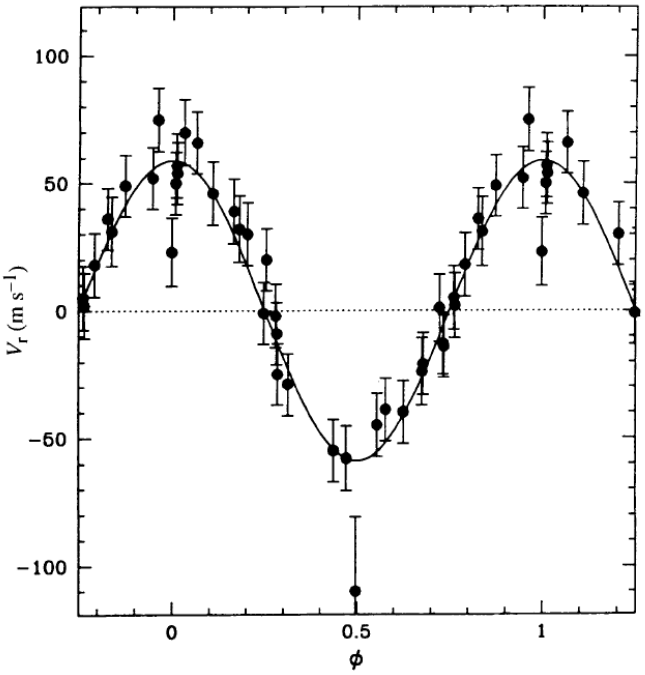


The use of Gaussian processes in the analysis of stellar noise in exoplanet search

João David Ribeiro Camacho
Mestrado em Engenharia Matemática
Departamento de Matemática
2017

Orientador
Nuno Miguel Cardoso Santos, Professor Auxiliar
Instituto de Astrofísica e Ciências do Espaço

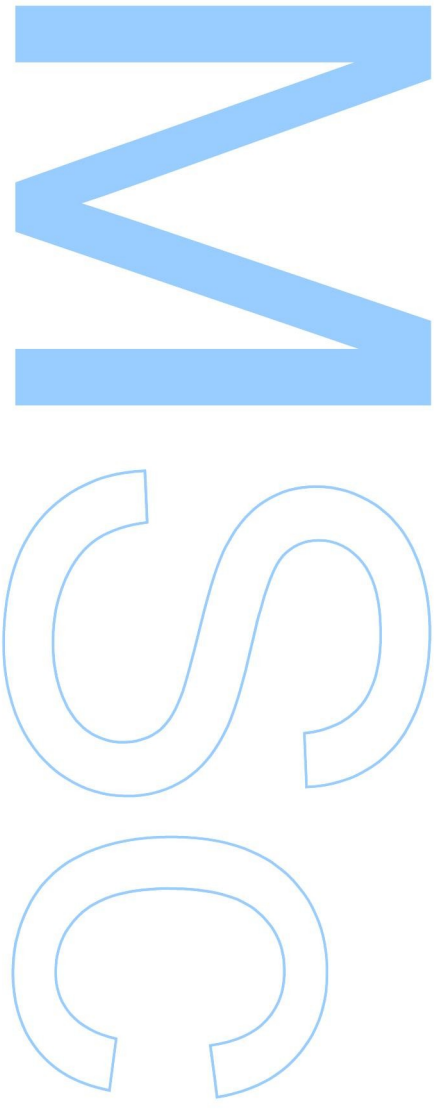
Co-orientador
João Pedro de Sousa Faria, Investigador
Instituto de Astrofísica e Ciências do Espaço

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,



Resumo

Uma quantidade significativa de tempo e de recursos tem sido dada à busca de planetas extra-solares nos últimos anos. Instrumentos como HARPS (High Precision Radial velocity Planet Searcher) deram aos astrónomos dados que ajudaram não só a moldar a estrutura da nossa vizinhança estelar, mas também a desenvolver uma melhor compreensão da evolução do Sistema Solar.

Isto foi possível graças ao uso de dados de velocidades radiais usado como um método indireto para encontrar exoplanetas. Infelizmente, ao analisar tais dados é importante ter em consideração que o sinal induzido por um planeta pode ser na realidade, por exemplo, sinais provenientes de manchas e flares solares. Tendo isto em consideração, o desenvolvimento de ferramentas capazes de analisar sinais de velocidades radiais e diferenciar com sucesso sinais estelares de sinais planetários é extremamente importante para a futura procura de planetas semelhantes à Terra.

Uma ferramentas possível para tal são os processos Gaussianos, que podem ser definidos como uma técnica não paramétrica usada em problemas de regressão e classificação. Estes processos governam as propriedades das funções e em vez de ajustar os parâmetros de uma base de funções, tentam inferir como todos os dados estão correlacionados.

Com base nessas características, foi desenvolvido um pacote para Python capaz de analisar dados reais usando processos Gaussianos. Com este pacote foram testados os limites desta abordagem na modelização de dados contaminados com ruído estelar de modo a encontrar a melhor maneira de analisar futuras medições de velocidades radiais de estrelas.

Palavras chave. Análise estatística: processos Gaussianos, exoplanetas: medições de velocidades radiais, exoplanetas: ruído estelar

Abstract

A significant amount of time and resources has been given to the search of extra-solar planets in recent years. Telescopes such as HARPS (High Accuracy Radial velocity Planet Searcher) have given astronomers valuable data that has helped not only shape our view of our stellar neighborhood, but also shape a better understanding of the evolution of the Solar System.

This was possible thanks to the use of precise radial velocity measurements that allow for an indirect method of finding exoplanets. Unfortunately, when analyzing such measurements it is important to take into consideration that the signal induced by a planet can be instead a signal of stellar origin, for example, originated by spots and plages. Taking this into consideration, the development of tools capable of analyzing radial velocities signals and successfully differentiate stellar signals from planetary signals is of great importance for the future search of earth-like planets.

One of such tools are Gaussian processes, that can be defined as a non-parametric technique mainly used in regression and classification problems. These processes governs the properties of functions, and instead of trying to fit the parameters of selected basis functions, try to infer how all the measured data is correlated.

Based on such characteristics, a Python package capable of analyzing real data using Gaussian processes was developed. The package tested the limits of this approach when fitting data contaminated with stellar noise in order to find the best way to analyze future radial velocities measurements of stars.

Keywords. Statistical analysis: Gaussian processes, exoplanets: radial velocity measurements, exoplanets: stellar noise

Acknowledgements

I would like to start by thanking my supervisors Nuno Santos and João Faria for accepting into their team and for all the help they gave me, time they have spent with me in this thesis, and that without it I would not be capable of accomplishing it.

I would also like to thank my mother and grandmother, that have always supported me and my dreams, without them I would have never been here in the first place, and I hope this work will make them proud.

Having said all that, there are a lot other people that can not be forgotten after all these years.

The 09s and its appendices (I'm looking at you Sandy and Cozido) is without a doubt the best year. Without their help and support, my life in Oporto would have been much more difficult, boring, and with far less alcohol.

The Micas e companhia Lda, Diana, Filipa, and Smurf showed me that mathematics students are not all that bad, many of them are actually very good friends that I will not forget, even if our future objective means that each one of us will follow different paths.

Finally there is someone I can't forget, and that made me the man I am today. To my dad.

Contents

1 Introduction **1**

1.1 Applications in Astronomy 1

1.2 Outline of this thesis 3

2 Gaussian processes **5**

2.1 Covariance Functions 6

2.1.1 Squared Exponential Kernel 8

2.1.2 Periodic Kernel 9

2.1.3 Rational Quadratic Kernel 9

2.1.4 Matérn family of kernels 11

2.1.5 White noise 12

2.1.6 Combining kernels 13

2.2 Gaussian process regression 15

2.2.1 Prediction with Gaussian processes 15

2.2.2 Kernels and their parameters 17

2.2.3 Parameter optimization 19

2.2.4 Markov Chain Monte Carlo 21

3 Doppler spectroscopy **23**

3.1 The radial velocity equation 23

3.1.1 Minimum mass 27

3.2 Radial velocities measurements 28

3.2.1 First successes 29

3.2.2 Instrumental limitations 30

3.3 Stellar noise contamination 31

3.3.1 Causes 31

3.3.2 Contamination examples 32

4 Gedi **35**

4.1 Introducing the Gaussian Jedi 35

4.1.1 Kernels 36

4.1.2 Log marginal likelihood 37

4.1.3 Optimization algorithms 37

4.1.4 Markov Chain Monte Carlo 38

4.2 Tests and performance 39

4.2.1 Covariance matrix calculation 39

4.2.2 Optimization 42

4.2.3 Markov Chain Monte Carlo 44

5 Results **47**

5.1 Data analysis 48

5.1.1 One spot analysis 49

5.1.2 Five spots analysis 53

5.1.3 Ten spots analysis 57

5.1.4 Twenty spots analysis 60

5.1.5 Radial velocity measurements of spots and a planet 64

6 Conclusions and future work **67**

6.1 Future work 68

Appendices **75**

A Algorithms for optimization **77**

B Gedi examples **81**

B.1 scipy.optimize 81

B.2 emcee 83

List of Figures

2.1 Comparison of the behavior of three squared exponential kernels of $\theta = 10$ 8

2.2 Effect of the length-scale on the ESS kernel 10

2.3 Comparison of the behavior of three rational quadratic kernels of $\theta = 10$ and $l = 1.0$, when fitting a random signal 10

2.4 Comparison of the behavior of the squared exponential kernel, exponential kernel, Matérn 32 kernel, and the Matérn 52 kernel 12

2.5 Samples of two white noise kernels 13

3.1 Description of the orbital motion elements of a planet or orbiting body around the center of mass of the system 24

3.2 Effects on the radial velocity signal that different values of e and ω produce, for a planet with the same mass and period 26

3.3 Diagram of confirmed exoplanets count as of 16 March 2017 given by the NASA Exoplanet Science Institute operated by the California Institute of Technology . 28

4.1 Comparison of the covariance matrix calculation for the squared exponential kernel, periodic kernel and the rational quadratic kernel. 40

4.2 Comparison of the covariance matrix calculation for the sum of the squared exponential kernel, periodic kernel and the rational quadratic kernel with a white noise kernel. 41

4.3 Comparison of the covariance matrix calculations done using the quasi-periodic kernels 41

4.4 Comparison of the time taken by `scipy.optimize` in the optimization of the squared exponential kernel, periodic kernel, and the rational quadratic kernel . . 43

4.5 Comparison of the time taken by `scipy.optimize` in the optimization of the sum of the squared exponential kernel, periodic kernel, and rational quadratic kernel with a white noise kernel 43
4.6 Comparison of the time taken by `emcee` running an MCMC with 1000 burn-ins and 2000 steps of the squared exponential kernel, periodic kernel, and rational quadratic kernel 45
4.7 Comparison of the time taken by `emcee` running an MCMC with 1000 burn-ins and 2000 steps of the sum of the exponential squared, sine squared exponential, and rational quadratic kernels with a white noise kernel 45

5.1 Log marginal likelihood obtained for each kernel while analyzing the RV signal of one spot with a measurement per day 51
5.2 Log marginal likelihood obtained for each kernel while analyzing the RV signal of one spot with a measurement every 4 days 53
5.3 Log marginal likelihood obtained for each kernel in the analysis of the radial velocity signals generated by five spots while having dataset with a measurement per day 55
5.4 Log marginal likelihood obtained for each kernel while analyzing the RV signal of five spots with a measurement every 4 days 56
5.5 Log marginal likelihood obtained for each kernel while analyzing the RV signal generated by ten spots having a measurement per day 58
5.6 Log marginal likelihood obtained for each kernel while analyzing the RV signal of ten spots with a measurement every 4 days 60
5.7 Log marginal likelihood obtained for each kernel while analyzing the RV signal generated by twenty spots having a measurement per day 62
5.8 Log marginal likelihood obtained for each kernel while analyzing the RV signal of twenty spots with a measurement every 4 days 63
5.9 Radial velocity measurements of the dataset created using the SOAP dataset of five spots with a linear decay, and the signal of a planet orbiting the star 64

List of Tables

5.1 MCMC results for the different analysis of a one starspot signal with the periodic kernel while having a measurement per day 50
5.2 MCMC results for the different analysis of a one starspot signal with the quasi-periodic kernel while having a measurement per day 50
5.3 MCMC results for the different analysis of a one starspot signal with the periodic kernel while having a measurement every four days 52
5.4 MCMC results for the different analysis of a one starspot signal with the quasi-periodic kernel while having a measurement every four days 52
5.5 MCMC results for the different analysis of five starspots signal with the periodic kernel while having a measurement every day 54
5.6 MCMC results for the different analysis of five starspots signal with the quasi-periodic kernel while having a measurement every day 54
5.7 MCMC results for the different analysis of five starspots signal with the periodic kernel while having a measurement every four days 56
5.8 MCMC results for the different analysis of five starspots signal with the quasi-periodic kernel while having a measurement every four days 56
5.9 MCMC results for the different analysis of ten starspots signal with the periodic kernel while having a measurement per day 58
5.10 MCMC results for the different analysis of ten starspots signal with the quasi-periodic kernel while having a measurement per day 58
5.11 MCMC results for the different analysis of ten starspots signal with the periodic kernel while having a measurement every four days 59
5.12 MCMC results for the different analysis of ten starspots signal with the quasi-periodic kernel while having a measurement every four days 59

5.13 MCMC results for the different analysis of twenty starspots signal with the periodic kernel while having a measurement every day 61
5.14 MCMC results for the different analysis of twenty starspots signal with the quasi-periodic kernel while having a measurement every day 61
5.15 MCMC results for the different analysis of twenty starspots signal with the periodic kernel while having a measurement every four days 62
5.16 MCMC results for the different analysis of twenty starspots signal with the quasi-periodic kernel while having a measurement every four days 63
5.17 MCMC results for the analysis of the dataset containing the radial velocity measurements of five spot with a linear decay, and the signal of a planet with the periodic kernel 65

Chapter 1.

Introduction

"Success is the ability to go from one failure to another with no loss of enthusiasm."

Sir Winston Churchill (1874-1965)

A challenge in science appears in how to perform the analysis of different types of data, and from it successfully determine relationships that further expand our knowledge of the environment around us. Whether one is a physicist or a chemist, a mathematician or a biologist, finding the best statistical tools to analyze a given problem is fundamental to obtain a significant conclusion, and help advance science.

One of such tools are known as Gaussian processes, named after the German mathematician Carl Friedrich Gauss, which are a powerful non-parametric tool with a long history in statistics. An early application of Gaussian processes can be seen in the work of Kriege (1951). In this work the author presented a statistical analysis of the behavior of gold value in mining. This inspired the work of Matheron (1962), were the basis of Kriging, also known as Gaussian process regression, was developed to be applied in geostatistics.

1.1 Applications in Astronomy

The search for extra solar planets, also called as exoplanets, is currently one of the most well known areas in Astronomy to the public. Every year numerous exoplanets and exoplanets candidates are announced, helping astronomers to improve the theories of formation and evolution of exoplanets (Adibekyan, Figueira, and Santos, 2016), and as a consequence helping improve the understanding of the formation and evolution of the Solar System.

To achieve that, astronomers have access to a wide range of methods to detect exoplanets including direct imaging, astrometry, Doppler spectroscopy, transit photometry and microlensing (Wright and Gaudi, 2013). Of all the available methods the two most successful so far proved to be the Doppler spectroscopy and transit photometry, with hundreds of exoplanets discovered by each of these two methods (see figure 3.3).

The detection by Mayor and Queloz (1995) of a planetary companion of the star *51 Pegasi*, Henry et al. (2000) and Charbonneau et al. (1999) detection of a planet around the star *HD 209458*, marked the first confirmed successes of the radial velocity method and the transit method respectively. Only in the recent year astronomers were capable of achieving a "boom" in exoplanet discoveries with the use of more precise instruments. With such increasing number of discoveries and available data to analyze, also came the need to find and/or develop the tools necessary to perform it and deal with the challenges presented by it.

Stellar noise produced by oscillations, granulations, and magnetic activity in the stars are capable of mimicking the signature of a planet in the radial velocity data (Dumusque et al., 2011a), thus difficulting the successful detection a extra-solar planet. A tool that seems to be capable of dealing with this issue is know as Gaussian processes used with success in regression and classification problems in other areas (Rasmussen and Williams, 2006).

Gaussian processes have already been used in Astronomy. One such use is in analyzing radial velocity measurements containing periodic signals caused by planets, even though such signal is contaminated with quasi-periodic stellar noise that can difficult the detection of Earth-like planets. For example, with them it was possible to estimate the mass of Kepler-78b applying Gaussian processes in radial velocity measurements from HIRES and HARPS-N spectrograph and Kepler photometry data (Grunblatt, Howard, and Haywood, 2015).

In the work developed by Rajpaul et al. (2015) a Gaussian Process framework was presented to model radial velocity time series to constrain and disentangle the stellar activity component in the radial velocity measurements from the planetary components, such framework was tested in three synthetic datasets, and although it depends on a number of approximations and empirical relationships, the authors were able to disentangle the signals from stellar activity and the presence of planet, even when the planetary signal was weaker and had a similar period to

the stellar activity signal. Unfortunately when the same framework was applied to a dataset publicly available of Alpha Centauri B they weren't successful in detecting a planetary signal and thus not able of confirming the presence of a planet as done by Dumusque et al. (2012).

Another example of the use of Gaussian processes can be seen in Brewer and Stello (2009) who described a method for inferring the frequencies and amplitudes of stellar oscillation modes from time-series observations of radial velocity data of stars that although computationally demanding was applied to the star ξ Hydrae. On it they had to take into account that the the predicted signature of an oscillation mode is not exactly sinusoidal and with a MCMC algorithm applied to two simulated data sets and concluded that although Gaussian processes applied on time series with 1500 to 15 000 points they were able to obtain results, for more than 15 000 points it stops being computationally feasible.

A more recent example is seen in the application of Gaussian processes to recover the orbits of both CoRoT-7b and CoRoT-7c from HARPS data (Haywood et al., 2014; Faria et al., 2016). In the later work, the authors used a Markov Chain Monte Carlo algorithm to infer the number of planets, from the available radial velocity measurements with no need for photometric observations to confirm the presence of such planets.

As such the use of Gaussian processes seems to have a promising future, especially considering that in the near future not only HARPS data will be available, but also data from ESPRESSO, a modern echelle sprectrograph built by the European Southern Observatory (ESO) in the Very Large Telescope (VLT) facility in the Atacama Desert, to search for exoplanets in our stellar neighborhood (Pepe et al., 2013).

1.2 Outline of this thesis

With the first light of ESPRESSO set for November of this year, it is fundamental to have appropriate tools capable of analyzing all the data that will become available. ESPRESSO will join other currently available spectrographs, such as HARPS and SOPHIE, in the search for Earth-like planets outside the solar system, and as consequence a careful and detailed analysis is important to obtain robust results.

As it is Gaussian processes have a promising future as one of the tools not to be ignored due to the results obtained so far. With that in mind is necessary to test the use of Gaussian processes in radial velocity measurements contaminated with stellar noise and learn the limitations of this method. How well this approach is in extracting physical information from contaminated data is important to establish, since only then we will be capable of distinguish stellar noise from a planetary signal.

Due to that, in chapter 2 we will give an explanation on how to work with Gaussian processes in the context of radial velocity measurements analysis, followed by chapter 3 where we discuss the characteristics of radial velocity measurements and how they are contaminated by stellar noise. In chapter 4 we present a package, written in Python, developed in this thesis, and that is freely available to use Gaussian processes to analyze radial velocity data. With chapter 5 were we present the results that can be obtained using the package developed, and what can be done in the future, to continue the work started in this thesis in chapter 6.

Chapter 2.

Gaussian processes

"The enchanting charms of this sublime science reveal themselves in all their beauty only to those who have the courage to go deeply into it."

Carl Friedrich Gauss (1777–1855)

Gaussian processes can be defined as a infinite-dimensional extension of normal random variables, making it one of the most advanced fields in statistics, presenting a wide range of tools for probabilistic modeling in various disciplines such as Finance, Data Mining, and Machine Learning (Lifshits, 2012). Currently they prove to be a important tool in statistics because it provides a promising Bayesian tool for modeling real-world statistical problems in a non-parametric model (Csató and Opper, 2001). On a Gaussian process the parameters we are trying to learn are functions, were we construct a prior distribution over functions and update it by conditioning the distribution to the data. As such and as explained by both Rasmussen and Williams (2006) and Robert and Casella (2004), Gaussian processes allow us to combine information brought by a sample function of our data with the prior information that is specified in a prior distribution, and summarize it in a posterior distribution.

To start talking about Gaussian processes and its most important properties it is required to first take a look into what a stochastic process is. We can say that a stochastic process $Y(\mathbf{x})$ is a collection of random variables indexed by $\mathbf{x}$ (Williams, 1998). From this brief definition we can expand it and say that a Gaussian process is a type of stochastic process, that can be defined as a collection of random variables, any of finite number of which have a joint Gaussian distribution (Rasmussen and Williams, 2006).

Having into account a process $f(\mathbf{x})$ we can interpret a Gaussian process as a generalization

of the Gaussian probability distribution and write it as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x'})), \tag{2.1}$$

from which it is necessary to define a *mean function* $m(\mathbf{x})$ and a *covariance function* $k(\mathbf{x}, \mathbf{x'})$, usually called as *kernel* for being a function of two arguments mapping a pair of inputs into $\mathbb{R}$.

Following the properties of a stochastic process we can easily define both as

$$\begin{aligned} m(\mathbf{x}) &= E[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x'}) &= E[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x'}) - m(\mathbf{x'}))], \end{aligned} \tag{2.2}$$

where E represents the expected value (Williams and Barber, 1998).

Any real-valued function $m(x)$ is in general acceptable to give rise to a valid Gaussian process (Do, 2008). As such, and for mathematical simplicity, it is usual to take the mean function as zero. The covariance function $k(\mathbf{x}, \mathbf{x'})$ on other hand must be chosen such that the resulting matrix be a valid covariance matrix, that is, a symmetric and positive semi-definite matrix, corresponding to some multivariate Gaussian distribution (Rasmussen and Williams, 2006).

These general properties imply of course, that the properties of a Gaussian process defined like this are controlled by the chosen kernel. This is a fact because Wilson (2014) concluded that it will be the kernel that will control the likely functions under a Gaussian process $f(\mathbf{x})$, and whether we wish to model a smooth function, periodic function, etc, to the data being analyzed.

2.1 Covariance Functions

An early objective in this thesis was to analyze what type of kernels would be better suited to work with, since it will be the kernel that controls all the modeling features when using Gaussian processes. Using Rasmussen and Williams (2006) as the main source of information for the initial exploration of the most important Gaussian processes properties, it was given a careful attention to *stationary kernels*, invariant to translations in the input space, as they are a function of $\mathbf{x} - \mathbf{x'}$, implying that the probability of observing a data-set will remain the same

if we move all the $\mathbf{x}$ by the same amount. Further more, we can say that if we have a kernel that is a function only of $|\mathbf{x} - \mathbf{x'}|$, it becomes invariant to all rigid motions, and becomes know as a *isotropic kernel*.

Given a kernel $k(\mathbf{x}, \mathbf{x'})$, and taking into consideration a set of input points $\{x_1, x_2, ..., x_n\}$, we are able to compute its *Gram matrix* K, given by

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \dots & k(x_n, x_n) \end{bmatrix}. \tag{2.3}$$

Such matrix is important when we are performing regression with Gaussian processes. Considering that k used in this analysis is a covariance function, instead of using the denomination of *Gram matrix*, it is more usual to call K simply as the *covariance matrix*.

One last aspect to have into consideration is that the real $n \times n$ matrix K is said to be positive semi-definite as it satisfies the condition

$$\mathbf{v}^T K \mathbf{v} \geq 0, \tag{2.4}$$

for all vectors $\mathbf{v} \in \mathbb{R}^n$ (Horn and Johnson, 2013). This property will become important to have into consideration later on, when it will be necessary to use the Cholensky decomposition to invert matrices.

Having into attention the basic characteristics defined previously, it was given attention in this thesis to a limited set of kernels, whose properties are explained and explored in the following pages. For this it was given extensively use to the works of Rasmussen and Williams (2006), Duvenaud (2014), and Wilson (2014) to understand and explain all the following kernels, and their basic properties. A more detailed analysis of the kernels and the full range of their properties is beyond the range of this thesis, and it is recommend those works.

2.1.1 Squared Exponential Kernel

The *squared exponential kernel*, also commonly called as *radial basis function* in the literature, and henceforth on referenced as the *ES kernel*, is one of the most studied and used kernels. If, for simplification of notation, we express the *ES kernel* as a function of $r = \mathbf{x} - \mathbf{x'}$, it can be written as

$$ES(r) = \theta^2 \exp\left(-\frac{r^2}{2l^2}\right). \tag{2.5}$$

The shape of this kernel is governed by two parameters, or sometimes also called *hyperparameters*, θ and l. More specifically the hyperparameter θ defines the amplitude of the kernel, and the hyperparameter l is the characteristic length-scale, which defines how smooth the kernel is. Small values of l imply that the kernel values are able to change quickly, while large values characterize a kernel whose value will change slowly, as it can be observed in figure 2.1.

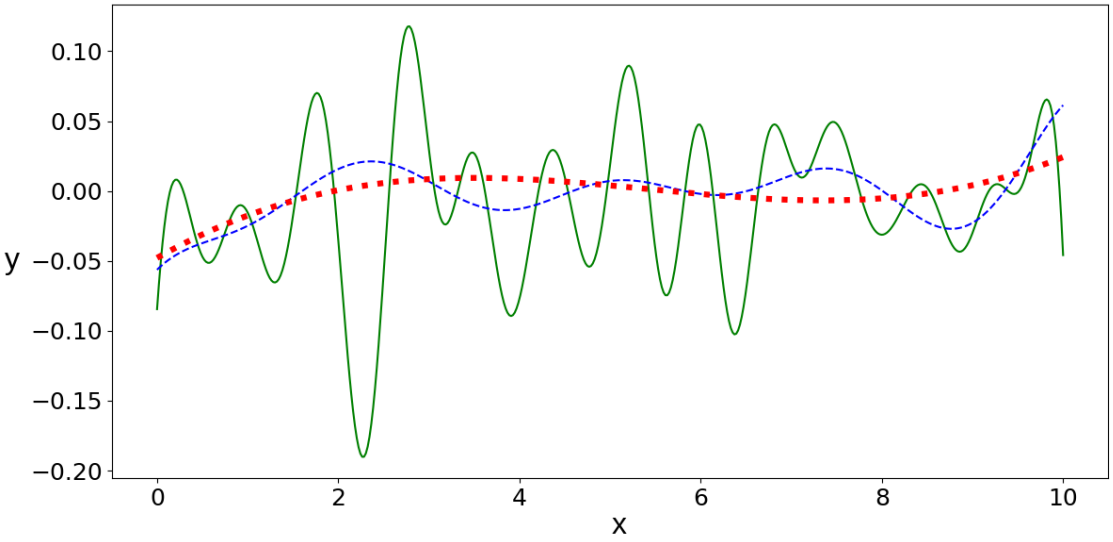


Figure 2.1: Comparison of the behavior of three squared exponential kernels of $\theta = 10$ fitting a sine-type signal similar to the one generated in B. At green we have a *ES kernel* with $l = 0.5$, at dashed blue $l = 2.0$, and at dotted red $l = 10.0$. On it it can be seen that as the characteristic lenght scale increases the functions generated will vary more slowly.

2.1.2 Periodic Kernel

The *periodic* or *exponential sine squared kernel*, and on this thesis referenced as *ESS kernel*, is one of the most (if not the most) useful kernels in this work. It is a strictly positive kernel that gives rise to periodic functions, most commonly used in combination with other kernels to model oscillatory data. It is expressed by

$$ESS(r) = \theta^2 \exp\left[-\frac{2}{l^2}\sin^2\left(\frac{\pi}{P}|r|\right)\right], \tag{2.6}$$

and characterized by three hyperparameters. As for with the *ES kernel*, the hyperparameter θ defines the amplitude of the kernel, and the hyperparameter l is the characteristic length scale. The new hyperparameter P on its turn, defines the periodic repetitions of the function, or in other words, the period often which the function repeats itself. This period P it is an important characteristic while analyzing radial velocity measurements, and thus making this kernel extremely useful, as it can identify periodic signals in the data being analyzed. In figure 2.2 we can observe how different values of the hyperparameters affect the behavior of the *ESS kernel*.

Another useful characteristic of this kernel is that when multiplied by the *ES kernel*, it creates what is know as *quasi-periodic kernel*. More will be said about this new kernel when we talk about combining kernels.

2.1.3 Rational Quadratic Kernel

The *rational quadratic kernel*, henceforth expressed simply as *RQ kernel*, is usually seen as an infinite set of *ES kernels* with different length scales, due to the fact that different mechanisms could be varying the data on different scales, an important aspect that can not be taken into consideration with the *ES kernel*.

This *RQ kernel* is expressed as

$$RQ(r) = \theta^2\left(1 + \frac{r^2}{2\alpha l^2}\right)^{-\alpha}, \tag{2.7}$$

where θ and l are defined in the same way as in the previous kernels and the hyperparameter

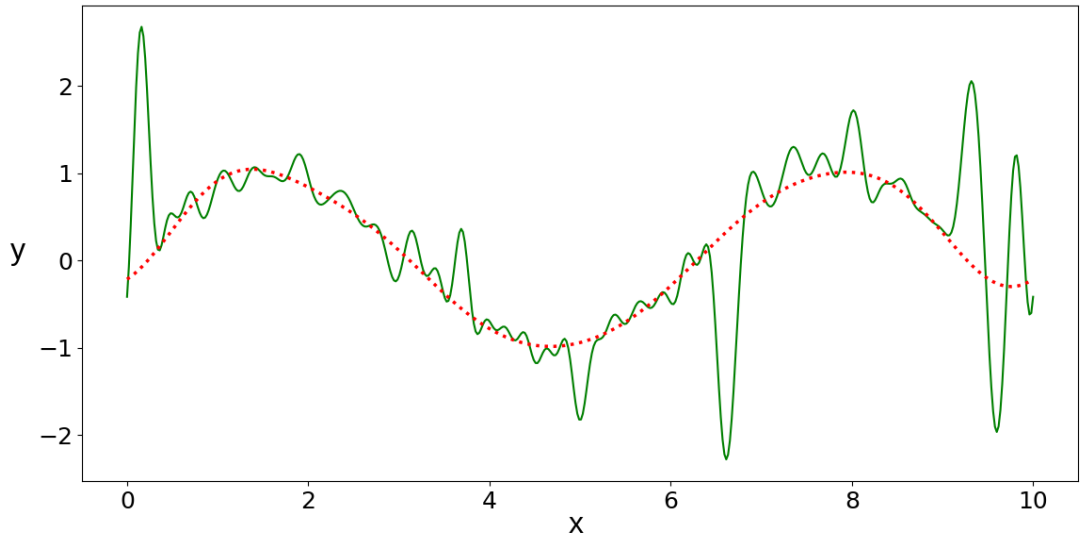


Figure 2.2: Effect of different length-scales on the fit of the *ESS kernel* on randomly generated sine-type signal. While both have a $\theta = 10$, and a $P = 10$, the lower length-scale the more wiggles a kernel will present, as it can be see in green with $l = 0.1$ and dotted red with $l = 100$, but without effecting the periodicity of the kernel.

α determines the weight of large and small scale variations. As $\alpha \to \infty$, it is possible to observe, although not shown in this thesis, that the *RQ kernel* converges to the *ES kernel* with characteristic length scale l. The influence of the hyperparameter α can be better observed in figure 2.3, where three different rational quadratic kernels are depicted.

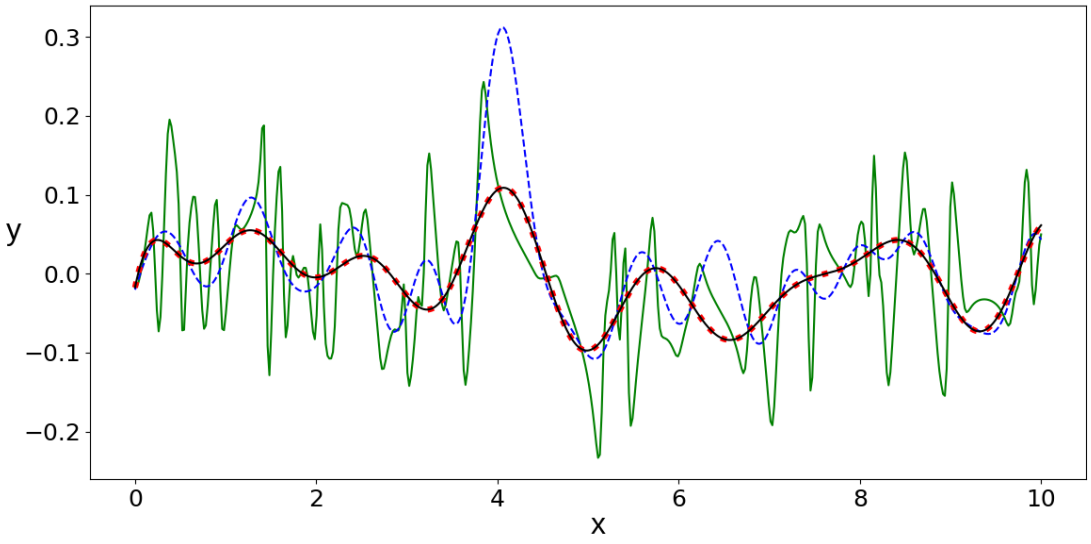


Figure 2.3: Comparison of the behavior of three rational quadratic kernels of $\theta = 10$ and $l = 1.0$, when fitting a random signal. In green we have a *RQ kernel* with $\alpha = 0.001$, dashed blue $\alpha = 1.0$, and dotted red $\alpha = 1000.0$. As α increases, the more similar the kernels tend to get to a *ES kernel*, with $\theta = 10$ and $l = 1.0$, in black.

2.1.4 Matérn family of kernels

The Matérn family or class of kernels, named after Bertil Matérn, is, as the *ES kernel*, one of the most popular kernels, and usually used as an alternative to that same kernel, that is formally expressed as

$$k(r) = \frac{2^{1-\upsilon}}{\Gamma(\upsilon)} \left(\frac{\sqrt{2\upsilon r}}{l} \right)^{\upsilon} K_{\upsilon} \left(\frac{\sqrt{2\upsilon r}}{l} \right), \tag{2.8}$$

where l is the characteristic length scale, K_{υ} is a modified Bessel function of second kind of order υ, $\Gamma(\upsilon)$ is the gamma function, and υ a positive parameter that becomes specially useful when it is set to a half-integer.

Resorting to expressions available in Abramowitz and Stegun (1972) it becomes possible to write equation 2.8 in order to υ half-integer values, that is not shown where due to not be fundamental to the work done in this thesis, but possible to be seen in Rasmussen and Williams (2006). From it is now possible to set the υ parameter to $^1/_2$, $^3/_2$, and $^5/_2$, that are the more interesting cases, extensively use in machine learning.

Setting the $\upsilon = {}^1/_2$, we are able to simplify equation 2.8 to a kernel known as the *exponential kernel* or *Ornstein-Uhlenbeck kernel*, and from here on labeled as *Exp kernel*, given by

$$Exp(r) = \theta^2 \exp\left(\frac{-|r|}{l} \right). \tag{2.9}$$

To equation 2.9 we have also included an amplitude hyperparameter θ, as it will be useful to the analysis made in chapter 2.2.

With the remaining two values ($\upsilon = {}^3/_2$ and $\upsilon = {}^5/_2$), it is possible to obtain the kernels usually know by *Matérn* $^3/_2$ and *Matérn* $^5/_2$ kernels, that for simplification of notation, are henceforth called as *M32* and *M52* kernels respectively given by

$$M32(r) = \theta^2 \left(1 + \frac{\sqrt{3}r}{l} \right) \exp\left(-\frac{\sqrt{3}r}{l} \right) \tag{2.10}$$

and

$$M52(r) = \theta^2 \left(1 + \frac{\sqrt{5}r}{l}\frac{5r^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}r}{l}\right). \tag{2.11}$$

This Matérn family of kernels is an alternative model recommended because of its general adoption, the *ES kernel* provides no flexibility with regards to a more local behavior, and essentially assume it is known a priori. In other words it will obtain a worse performance if the data varies drastically from one point to the next, unlike the Matern family (Stein, 1999). From figure 2.4 it is possible to compare the behavior of the Matérn kernels and the *ES kernel*.

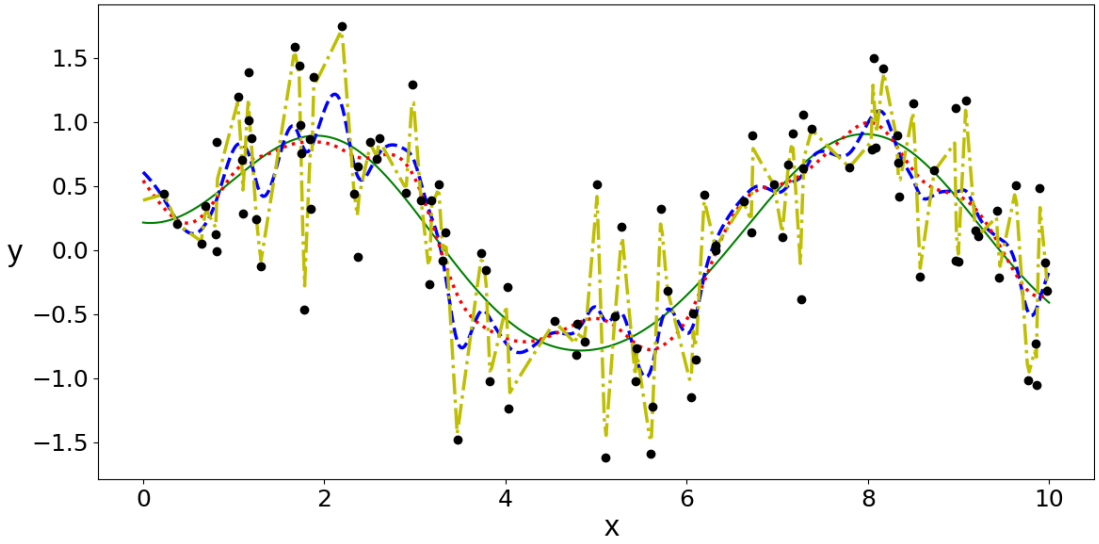


Figure 2.4: Comparison of the behavior of the *ES kernel* (green line), *Exp kernel* (yelow dotted and dashed line), *M32 kernel* (dashed blue line), and *M52 kernel* (dotted red line). All with $\theta = 10$ and $l = 1.0$ fitting a sine wave with jitter. On it is possible to observe that to an equal set of parameters, the Matérn kernels show in fact, to be more flexible to the "irregularities" of the data.

2.1.5 White noise

Another well known and important kernel necessary to define is the *white noise kernel*, or as it will be called in this thesis, *WN kernel*, that can be defined as

$$WN(r) = \theta^2 \delta_{ij} r \tag{2.12}$$

where θ defines the amplitude as usual, and δ_{ij} is know as the Dirac delta function that for a

given set of (x_i, x_j) is

$$\delta_{ij} = \begin{cases} 1, & \text{if } x_i = x_j, \\ 0, & \text{if } x_i \neq x_j. \end{cases} \tag{2.13}$$

The *WN kernel* importance arise in following subsection, where we perform the sum of two kernels. The *WN kernel* is used to explain the noise component of a set of data, and its hyperparameter θ will give the noise level of the data. In figure 2.5 we are able to observe the behavior of white noise of different amplitudes.

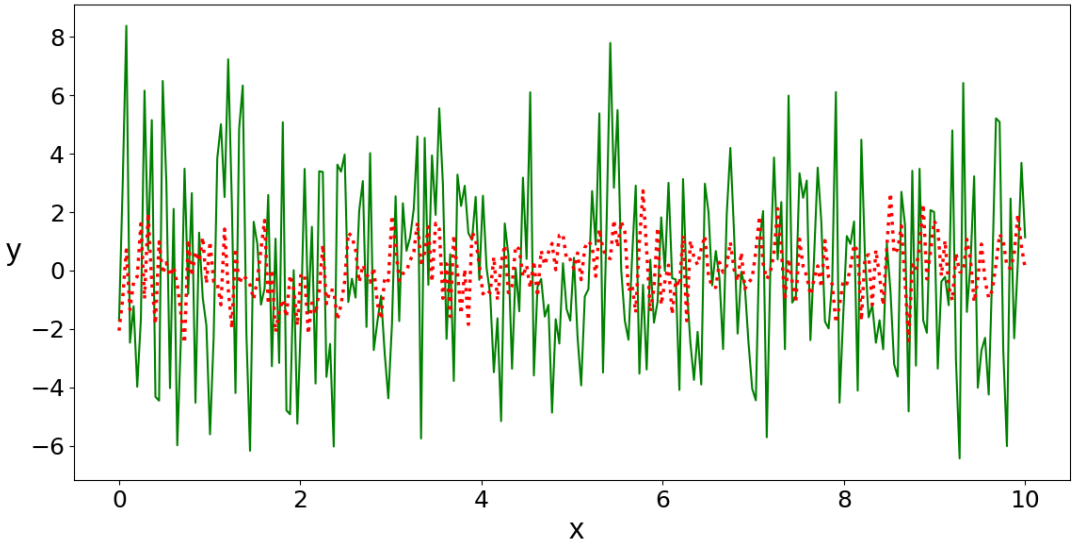


Figure 2.5: Comparison of the behavior of the *WN kernel* with different amplitudes, in red dashed line samples drawn with $\theta = 1$ and in green line samples with $\theta = 10$.

2.1.6 Combining kernels

Although the previous kernels are capable of expressing interesting characteristics on the data being analyzed, they alone are are not very useful, or have problems defining all the characteristics we want. The most common process to solve such issue is to use different types of kernels, either by addition and/or multiplication, to express more complex characteristics observed in the data.

Both addiction and multiplication of two given kernel, will generate a new kernel that can

be quite easily represented as

$$\begin{aligned} k_{new}(r) &= k_1(r) + k_2(r) \\ k_{new}(r) &= k_1(r) \times k_2(r). \end{aligned} \tag{2.14}$$

As stated previously, the addiction of the *WN kernel* with, for example, the *ES kernel*, will allow us to model the noise component of the data, although in general, there isn't a clear distinction between signal and noise.

Another important combination that was already mentioned is the *quasi-periodic kernel* that is formed by the product of the *ES kernel* with the *ESS kernel*. This combination can be expressed as

$$ES(r) \times ESS(r) = {\theta_1}^2 \exp\left(-\frac{r^2}{2{l_1}^2}\right) \times {\theta_2}^2 \exp\left[-\frac{2}{{l_2}^2}\sin^2\left(\frac{\pi}{P}|r|\right)\right], \tag{2.15}$$

and be simplified as

$$ES(r) \times ESS(r) = \theta^2 \exp\left[-\left(\frac{2}{{l_1}^2}\sin^2\left(\frac{\pi}{P}|r|\right) + \frac{r^2}{2{l_2}^2}\right)\right]. \tag{2.16}$$

Like in previous kernels θ will give the amplitude of the signal, both l_1 and l_2 are characteristic length scales, that we can associate to a periodic length scale and aperiodic length scale respectively, accordingly to the term in the equation they are in, helping defining the smoothness of our kernel. For last we have P that represents the quasi-periodicity of the kernel.

With this combination we are capable of modeling a periodic function whose shape changes over time, for example either decreasing or increasing its amplitude. While its shape doesn't repeat exactly over time, looking at the data in a smaller timescale it has a more local periodicity.

2.2 Gaussian process regression

It is usual to use Gaussian processes to solve two types of problems. A classification problem or to solve a regression problem, depending on the type of output we will be dealing with. Gaussian process regression can be seen as defining a distribution over functions, with the inference taking place in the space of functions (Rasmussen and Williams, 2006). With it, we can say that we are nothing more than performing non-linear interpolation in which the interpolated values are modeled using a Gaussian process.

2.2.1 Prediction with Gaussian processes

A good way to start talking about Gaussian process regression is determining how prediction is made. To a given set of observation $\mathbf{y} = y_1, y_2, ..., y_n$ made at certain instances $\mathbf{x} = x_1, x_2, ..., x_n$, how can we make our best estimate for $\mathbf{f}$ at x_{n+1} or any other value $x_\star$?

Following Williams and Barber (1998) and Rasmussen and Williams (2006), we can say that a prior over functions allow us to make our prediction $y_\star$ of the expected value at $x_\star$, and represent it as

$$\begin{bmatrix} \mathbf{y} \\ f_\star \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(\mathbf{x},\mathbf{x}) & K(\mathbf{x},x_\star) \\ K(x_\star,\mathbf{x}) & K(x_\star,x_\star) \end{bmatrix}\right), \tag{2.17}$$

where $K(\mathbf{x},\mathbf{x})$ denotes the $n \times n$ covariance matrix of all pairs in $\mathbf{x}$, similarly, from $x_\star$ and $\mathbf{x}$ we can obtain the entries $K(\mathbf{x},x_\star)$, $K(x_\star,\mathbf{x})$, and $K(x_\star,x_\star)$. Making the same simplification of notation as Rasmussen and Williams (2006), from here on we will represent $K = K(\mathbf{x},\mathbf{x})$ and $K_\star = K(\mathbf{x},x_\star)$ and for the case of only having one test point $x\star$, we will write it as $k_\star$.

If we denote the prior over functions as $P(\mathbf{y})$, and in the same logic, denote the prior $P(y_\star,\mathbf{y})$ as the joint distribution that includes $y_\star$, and the probability of observing the values of $y_1, y_2, ..., y_n$ at $x_1, x_2, ..., x_n$ as $P(\mathbf{x}|\mathbf{y})$, we are able to write

$$P(y_\star|\mathbf{x}) = \int P(y_\star,\mathbf{y}|\mathbf{x})d\mathbf{y}. \tag{2.18}$$

If we now have into consideration Bayes' theorem (Sivia and Skilling, 2006)

$$P(A|B) = \frac{P(B|A)P(B)}{P(B)}, \text{ if } P(B) \neq 0 \tag{2.19}$$

where $P(A)$ and $P(B)$ are the probabilities of observing the events A and B, respectively. $P(A|B)$ is the conditional probability of observing the event A given that the event B was observed, and $P(B|A)$ is the conditional probability of observing the event B given that the event A was observed (Orloff and Bloom, 2014).

We can now rewrite equation 2.18 as

$$P(y_\star|\mathbf{x}) = \frac{1}{P(\mathbf{x})} \int P(y_\star|\mathbf{y})P(\mathbf{y})P(\mathbf{x}|\mathbf{y})d\mathbf{y}, \tag{2.20}$$

that allow us to have in the end

$$P(y_\star|\mathbf{x}) = \int P(y_\star|\mathbf{y})P(\mathbf{y}|\mathbf{x})d\mathbf{y}, \tag{2.21}$$

meaning that the predictive distribution of $y_\star$ is found from the marginalization of the product of the prior and the model. If $P(\mathbf{x}|\mathbf{f})$ and $P(y_\star|\mathbf{f})$ are Gaussian, we will have that $P(y_\star|\mathbf{x})$ is also Gaussian with mean and variance calculated using matrix computations with matrices of size $n \times n$.

Deriving the conditional distribution, we arrive at

$$f_\star|\mathbf{x}, \mathbf{f}, x_\star \sim \mathcal{N}(\bar{f}_\star, cov(f_\star)), \tag{2.22}$$

from which we can obtain the mean and variance expressions (see Rasmussen and Williams (2006)), respectively given by

$$\bar{f}_\star = \mathbf{k}_\star^\intercal K^{-1}\mathbf{y}, \tag{2.23}$$

$$\mathrm{V}[f_\star] = k(x_\star, x_\star) - \mathbf{k}_\star^\intercal K^{-1}\mathbf{k}_\star, \tag{2.24}$$

making now possible to predict a estimate for $f_\star$ at value $x_\star$.

2.2.2 Kernels and their parameters

To a given kernel it is relatively easy to make a prediction on a new test point. Unfortunately in practical situations it is unlikely that we will know which kernel or set of kernels are best to be used, and searching for the best solution by trial and error turns out to be impractical (Williams and Barber, 1998).

As such, it is now useful to explain the concept of *marginal likelihood*. Rearranging equation 2.21 it is possible to express the marginal likelihood as

$$P(\mathbf{y}|X) = \int P(\mathbf{y}|\mathbf{f})P(\mathbf{f}|X)d\mathbf{f}, \tag{2.25}$$

or in other words, it is possible to express it as the integral of the likelihood times the prior.

From a more complete deduction that can be seen in Rasmussen and Williams (2006), we are able to perform the integration of equation 2.25, and from it obtain the *log marginal likelihood* given by

$$\log P(\mathbf{y}|X,\theta) = -\frac{1}{2}\mathbf{y}^\intercal K^{-1}\mathbf{y} - \frac{1}{2}\log|K| - \frac{n}{2}\log(2\pi), \tag{2.26}$$

where K is the covariance matrix, $\mathbf{y}$ is the points, n is the number of points, and θ simply represents the parameters of the kernel in use.

This value is useful when comparing different models. The higher the value the better. For example, a model whose log marginal likelihood is -1 is better to a model whose log marginal likelihood is -10.

The calculation of log marginal likelihood presented in equation 2.26 can still be interpreted by its three terms. The term $-\frac{1}{2}\mathbf{y}^\intercal K\mathbf{y}$, is the only one that includes the data $\mathbf{y}$, the second term $-\frac{1}{2}\log|K|$ is the complexity penalty that depends on the kernel being used, and last the term $\frac{n}{2}\log(2\pi)$ is a normalization constant.

Having the log marginal likelihood defined by equation 2.26, it is also useful to obtain the

partial derivatives of the log marginal likelihood with respect to the parameters that can be written as

$$\frac{\partial}{\partial\theta}\log P(\mathbf{y}|X,\theta) = \frac{1}{2}\mathbf{y}^{\intercal}K^{-1}\frac{\partial K}{\partial\theta}K^{-1}\mathbf{y} - \frac{1}{2}tr\left(K^{-1}\frac{\partial K}{\partial\theta}\right). \tag{2.27}$$

Equation 2.27 allow us now to search for the set of parameters that optimize the marginal likelihood, for example with the use of a gradient descent algorithm. Such optimization processes will be better explained in a the next subsection, and as such we will end its discussion here.

Having the previous equations into consideration, and before entering the optimization routines that can be used, we can now question ourselves on how much computation power is required to perform a Gaussian process regression?

The main concern in the Gaussian processes computation is on the $n \times n$ covariance matrix, as it depends on the observed inputs and the values of the parameters of the kernels. The difficulty of the computation of the covariance matrix arise due to its condition number, that is the ratio of its largest eigenvalue to its smallest eigenvalue (Neal, 1998). If it ends up having a large condition number, round-off error in the computations may cause the matrix inversion to fail and/or be inaccurate.

An easy way to solve such problem relies on including noise as it contributes additively to all eigenvalues of the covariance matrix. Such addition is particular useful to avoid having a covariance matrix that is not positive-definite, as it would make the use of the Cholensky factorization unpractical. The Cholesky factorization is a useful algorithm that help us rewriting the covariance matrix as

$$K = LL^{\intercal}, \tag{2.28}$$

where L is the lower triangular matrix of K, and whose primary use is to compute the inverse of the covariance matrix K^{-1}, since the computation on the inverse matrix of K may be sometimes

impractical to perform.

Another simplification that can be perform on equation 2.26 relies on that, if we consider that the determinant calculation of a positive definite symmetric matrix is given by

$$|K| = \prod_{i=1}^{n} L_{ii}^2, \tag{2.29}$$

then, considering simple logarithmic properties, we can write

$$\begin{aligned} \log|K| &= \log\left(\prod_{i=1}^{n} L_{ii}^2\right) \\ &= 2\log\left(\prod_{i=1}^{n} L_{ii}\right) \\ &= 2\log\left(L_{11} \times L_{22} \times \cdots \times L_{nn}\right) \\ &= 2\left(\log L_{11} + \log L_{22} + \cdots + \log L_{nn}\right) \\ &= 2\sum_{i=1}^{n} \log L_{ii}. \end{aligned} \tag{2.30}$$

This simplification proves useful when the covariance matrix becomes large (Rasmussen and Williams, 2006).

Having equations 2.28 and 2.30, it is now possible to rewrite equation 2.26 as

$$\log P(\mathbf{y}|X,\theta) = -\frac{1}{2}\mathbf{y}^\intercal L^\intercal L\mathbf{y} - \sum_i \log L_{ii} - \frac{n}{2}\log(2\pi), \tag{2.31}$$

that is computational less demanding. The same substitutions done previously can still be consider useful do be done in equations 2.23, 2.24 and 2.27, diminishing even more the computational costs when performing Gaussian process regression.

2.2.3 Parameter optimization

When talking about Gaussian process regression, our main objective is finding the best hyperparameters of a given kernel taking into consideration the data being analyzed. A common technique to find the optimal values of the hyperparameters rely into taking advantage of the gradient of the hyperparameters that characterize the kernel and that can be obtain by equation 2.27.

The gradient is a useful tool to find the optimal values because, if we consider the gradient of a function given by

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right), \tag{2.32}$$

and if we move along the gradient direction, we will be moving into the direction on which the function value increases at the fastest rate, or as it is usual called, the direction of steepest ascent (Rao, 2009). Having this property into consideration, if we can take the negative of the gradient vector we walk into the direction of steepest descent, or in simpler words, walk in the direction of the minimum of the function.

Using this basic property, the first method implemented in the work done in this thesis was the *steepest descent algorithm*, or as it is also know, *gradient descent algorithm*. Given a initial point, we iteratively move along the steepest descent direction according to selected steps, until the optimal value is met, given a pre-determined set of stopping criteria. A more detailed explanation of this algorithm can be seen in Rao (2009) and a pseudo-algorithm of it can be seen in algorithm 1 of the appendix.

Another algorithm implemented is one of the most popular quasi-Newton algorithms (Nocedal and Wright, 2006). It is know as *Broyden-Fletcher-Goldfarb-Shanno algorithm*, or simply as *BFGS algorithm*, that makes use of the Hessian matrix, updates it iterativelly, and exhibits super-linear convergence near the optimal point (Rao, 2009). Once again a pseudo-algorithm is shown on the appendix that allows for a better understanding of the algorithm's logic in algorithm 2.

The last algorithm implemented in this thesis has its origin in a algorithm know as *RPROP*, also known as *resilient back propagation algorithm* proposed by Riedmiller and Braun (1993). The *RPROP algorithm* is a not very known first-order optimization algorithm which can adapt the step length based on the sign the gradient. In contrast the steepest descent, takes iterative steps proportional to the negative local gradient (Blum and Riedmiller, 2013).

In this algorithm, the correct step length, for a given parameter, at iteration X_{i+1} is determined by the information of the gradients at X_i and X_{i+1}, instead of performing a line-search

for the optimal step length as it usually happens with other algorithms. If the gradients at X_i and X_{i+1} change of sign then the local minimum had been jumped over and is required to redo the calculation with a smaller step length, otherwise the step length could be increased to speed up the convergence. The paper of Riedmiller and Braun (1993) recommended that if the sign had change the step should be decreased by 0.5, and if the sign had not changed the step should be increased by 1.2 in order to improve convergence time.

Based on the logic behind the *RPROP algorithm* and the easy and simple implementation of the *steepest descent algorithm*, it was combined the properties of this two algorithms into the algorithm 3 of the apendix, called here as *alternative steepest descent algorithm* or *altSDA* for short.

2.2.4 Markov Chain Monte Carlo

Although the usage of gradient based algorithms to optimize the kernel's hyperparameters proves to work, their efficiency, most of the time, is very sensitive to the choice of initial guess made for the parameters. To have a good result it is required to the user to have a good initial idea of what should be the optimal parameters. Otherwise we might get the algorithm stuck in a local minimum instead of finding the global minimum.

A way of solving this problem is to use Markov chain Monte Carlo (MCMC) algorithms with our Gaussian process. In the work of Titsias, Rattray, and Lawrence (2011) we are able to find a good explanation on how to apply MCMC to Gaussian processes, and to follow a full Bayesian approach to sample the kernel's parameters.

We can use a random walk Metropolis-Hastings algorithm to simulate a proposal X_{t+1} given X_t from

$$X_{t+1} = X_t + \epsilon_t, \tag{2.33}$$

where ϵ_t is a random perturbation with a symmetric distribution (Robert and Casella, 2004).

With it we are able to use a acceptance distribution A given by

$$A = min\left(1, \frac{p(y|X_{t+1})p(X_{t+1})Q(X_t|X_{t+1})}{p(y|X_t)p(X_t)Q(X_{t+1}|X_t)}\right), \tag{2.34}$$

that can be simplify due to our proposal density being symmetric, turning our acceptance distribution into a simple likelihood ratio. With this simple algorithm, we are able to create a alternative to the optimization developed in the previous subsection for our Gaussian processes.

Chapter 3.

Doppler spectroscopy

"Take comfort, the time will come when all men will see as I do."

Giordano Bruno (1548-1600)

The search for extra-solar planets, also known as exoplanets, is one of the most exciting and rapidly evolving areas of Astronomy. With several exoplanets candidates announced every year, and dozens of them being located in the so called habitable zone of its star, making it possible to have water in liquid state, not only allowed astronomers to be closer to find a Earth-like planet, but also help them to improve the theory behind the formation and evolution of exoplanets, improving as a consequence, the theory of how our own planet was formed (Adibekyan, Figueira, and Santos, 2016).

One of the most successful methods to discover extra solar planets is the Doppler spectroscopy, more commonly known as the radial velocity method, with more than 500 confirmed exoplanets discovered since the milestone discovery of *51 Pegasi b* by Mayor and Queloz (1995).

In this method the planet is indirectly observed by the change of the radial velocity of the star along its orbit around the center of mass of the star-planet system. From this effect it is possible to obtain information about the planet's period, distance, shape of the orbit, and an estimate of its mass (Udry and Santos, 2007).

3.1 The radial velocity equation

To understand how such information is obtained from Doppler spectroscopy it is better to first understand the orbital properties of a planetary system observed when using this method.

The orbital concepts and properties that will be explained in the following pages can be seen in figure 3.1, taken from the work of Perryman (2011), were a simple scheme of the orbit of a planet and a star around the center of mass of the system is represented.

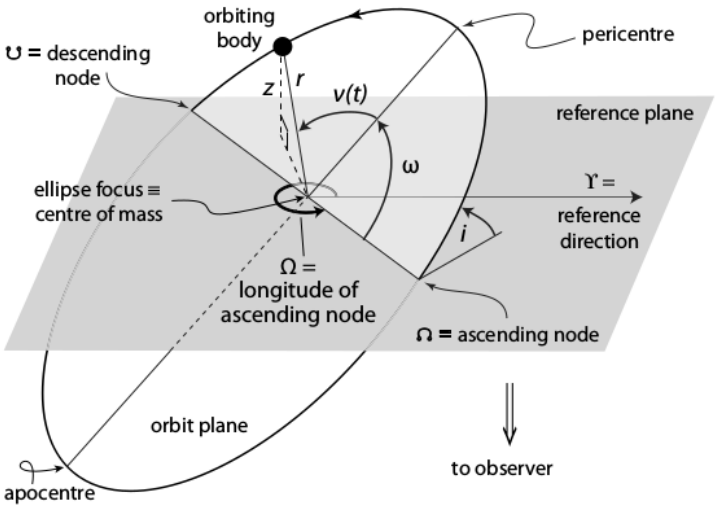


Figure 3.1: Description of the orbital motion elements of a planet or orbiting body around the center of mass of the system. On it we can observe, among other things, the argument of periastron ω measured from the ascending node and the true anomaly, here represented as $\nu(t)$, that is measured with respect to the periastron. Source: Perryman (2011).

To start our analysis, one of the most important aspects to have in mind are the Kepler's laws that can be quickly summarized into

1st - All planets move in elliptical orbits with the Sun occupying one of focus.

2nd - The line that connects a planet to the Sun sweeps equal areas in equal times.

3rd - The squares of the periods of the planets are proportional to the cubes of its semi-major axis of its orbits.

It can be seen that Kepler's third law can be express by

$$P^2 = \frac{4\pi^2}{GM} a^3, \tag{3.1}$$

where P is the period of the planet, a is the semi-major axis the orbit, G is the gravitational constant, and M is the mass of the star (Perryman, 2011).

Depending on the type of orbit that is being measured equation 3.3 can become

$$P^2 = \frac{4\pi^2}{G(M_\star + M_p)} a_{rel}^3, \tag{3.2}$$

if we are having into consideration the relative orbits, or in other words, where the planet's motion is measured relatively to the star rather to the center of mass of the system. Having this consideration in mind, we have $M_\star$ and M_p as the masses of the star and the planet, respectively, and a_{rel} as the semi-major axis of the relative orbit.

If we instead take the absolute orbits, that is the orbit of the star around the center of mass of the planetary system, we can then write equation 3.3 as

$$P^2 = \frac{4\pi^2}{G}\frac{(M_\star + M_p)^2}{M_p^3}a_\star^3, \tag{3.3}$$

where $a_\star$ represents the semi-major axis of star orbit around the center of mass.

With this two types of orbits considered, we can take into consideration the equation of the ellipse, in polar coordinates, of one of the bodies of the system around the center of mass

$$r(1 + e\cos\nu) = a(1 - e^2), \tag{3.4}$$

with ν representing the true anomaly, e the eccentricity of the orbit, and r the distance of the body to the center of mass (Wright and Gaudi, 2013). The computations of the body's position in its orbit, can be performed with the use of the eccentric anomaly E that is related with the time t since the star passed at periastron T, through the mean anomaly M given by

$$M = \frac{2\pi(t - T)}{P} = E - e\sin E. \tag{3.5}$$

This relation can be derived from orbital dynamics (Perryman, 2011) and allow us to obtain the true anomaly given by

$$\tan\left(\frac{\nu}{2}\right) = \sqrt{\frac{1+e}{1-e}}\tan\left(\frac{E}{2}\right). \tag{3.6}$$

With the orbital properties of a planetary system introduced, we can now present the radial velocity equation as given by Perryman (2011),

$$v_r = K[cos(\omega + \nu) + e\cos(\omega)]. \tag{3.7}$$

Such equation gives us the projected motion of the star around the center of mass of the planetary system. For it we need to have into consideration the true anomaly ν, the argument of the periastron ω, and the radial velocity semi-amplitude K that will be better expressed soon.

For the sake of consistency, we can still add a term γ to equation 3.7 and express it as

$$v_r = K[cos(\omega + \nu) + e\cos(\omega)] + \gamma, \tag{3.8}$$

to adjust the movements of the star in the galaxy, that is, the systematic velocity of the star in relation to the solar system.

Having different values for ω, e, and ν we will obtain different radial velocity curves, as seen in figure 3.2.

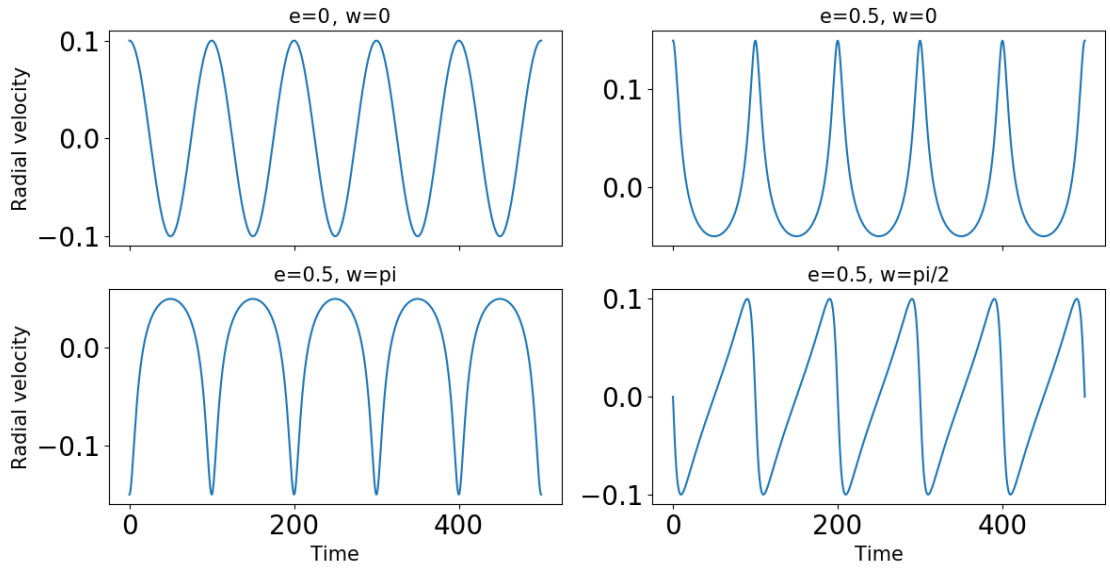


Figure 3.2: Effects on the radial velocity signal that different values of e and ω produce, for a planet with the same mass and period. The eccentricity of the orbit and the argument of the periastron of a extra-solar planet clearly influence the expected signal.

From either equations 3.7 and 3.8 the radial velocity semi-amplitude can be expressed as

$$K = \frac{2\pi}{P}\frac{a_\star\ sin(i)}{\sqrt{1-e^2}}, \tag{3.9}$$

where the new term i represents the inclination of the orbit relative to the reference plane (see figure 3.1).

3.1.1 Minimum mass

Combining equation 3.9 with equation 3.2, it is possible to rewrite K as

$$K^2 = \frac{G}{1-e^2}\frac{1}{a_\star \sin(i)}\frac{M_p^3 \sin^3(i)}{(M_\star + M_p)^2}. \tag{3.10}$$

From equation 3.10 we can now look at the last fraction, called mass function $\mathcal{M}$, derived from the orbital period and radial velocity of the planet obtained through radial velocity measurements

$$\mathcal{M} = \frac{M_p^3 \sin^3(i)}{(M_\star + M_p)^2}, \tag{3.11}$$

and from it, we can have the minimum mass M_{min}

$$M_{min} = M_p \sin(i), \tag{3.12}$$

that is a good approximation for the true mass of the planet.

Having the minimum mass into consideration, and that in general we have $M_\star >> M_P$, it is thus possible to rewrite equation 3.10 into

$$K = \left(\frac{2\pi G}{P}\right)^{1/3} \frac{M_{min}}{M_\star^{2/3}}(1-e^2)^{-1/2}. \tag{3.13}$$

Although the radial velocity method has limitations in term of what we can learn from the planetary system. The minimum mass M_{min} and the orbital parameters given by this method allows us to combine its data with others, such as, the transits method to help astronomers to have a better understanding of a extra solar planet. For example, with the estimation of the minimum mass given by the radial velocity method, and the planet's radius given by the transit method, we are able to determine the mean density of an exoplanet (Rice, 2014).

3.2 Radial velocities measurements

One of the early proposals to use radial velocities measurements as a mean to search for extra-solar planets was made by Struve (1952). He proposed that using the using the most powerful spectrographs in existence at the time, it would be possible to detect a planet with the mass of Jupiter orbiting the star at a distance of $^1/_{50}\ AU$, as this would create a radial velocity oscillation in the star of approximately $\pm 0.2 km/s$. In the following decades the improvements done in the instruments made possible to achieve precisions of $15 m/s$ in the 1990's by the time of the discovery of the first exoplanet using radial velocity measurements by Mayor and Queloz (1995) .

As it can be seen in figure 3.3, the radial velocity technique has proved useful in the detection of high mass, Jovian like planets, capable of causing the strongest gravitational perturbation on its host star, and as such, the greatest changes in the star radial velocity. The radial velocity method is, in many cases the only way of measuring the mass of a planet, making it an important to help confirming and characterizing planets detected with other methods.

More and more Neptune-type planets and Super Earths have also been observed recently, while high accuracy spectrographs try to overcome the technical and physical problems of detection smaller radial velocities and try to achieve the milestones of discover Earth-like planets.

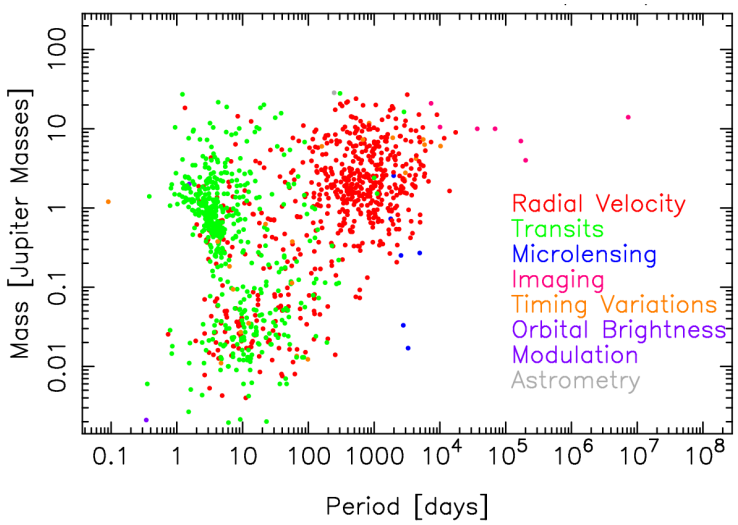


Figure 3.3: Confirmed exoplanets count as of 16 March 2017 given by the NASA Exoplanet Science Institute operated by the California Institute of Technology. As it is observable, the radial velocity method proved so far quite useful in the detection of planets of Jupiter mass planets.
Source: `http://exoplanetarchive.ipac.caltech.edu/exoplanetplots/`

3.2.1 First successes

The spectrograph evolution achieved in the second part of the 20th century culminated with the discovery of *51 Pegasi b* by Mayor and Queloz (1995) using the fiber-fed echelle spectrograph ELODIE of the Haute-Provence Observatory. With this spectrograph it had been possible to achieve an accuracy around $13m/s$, that made possible to indirectly detect the radial velocity signature of a planet with a minimum mass (M_{min}) of $0.47 \pm 0.02 M_{Jupiter}$ and a period (P) of 4.2293 ± 0.0011 days (Mayor and Queloz, 1995).

The peculiar characteristic of *51 Pegasi b* was followed by controversy but it did not stop the discovery of two new planets in 1996. Marcy and Butler (1996) discovered one around the star *70 Virginis* with $M_{min} \approx 6.6 M_{Jupiter}$ and $P \approx 116.6$ days. By its turn Butler and Marcy (1996) discovered another around the star *47 Ursae Majoris* with $M_{min} \approx 2.39 M_{Jupiter}$ and $P \approx 1090$ days.

Even with several detections of exoplanets using radial velocity measurements the community was not convinced that the discovery of objects with short period, know today as *hot Jupiters* were indeed planets (Mayor, Lovis, and Santos, 2014). The models of planet formation expected that giant planets would be formed in the regions beyond the ice-line, but it was shown that planet migration was possible, and that *51 Pegasi b* was indeed a planet that formed at a distance of 5 AU of its host star, and migrated inward through interactions with the circumstellar disk (Lin, Bodenheimer, and Richardson, 1996).

It was necessary to wait until the year 1999 for the discovery of the first planet with the transit method by Charbonneau et al. (1999). This milestone was fundamental, as it was a short period gas giant planet around the star *HD 209458* with a radius of $1.27 \pm 0.02 R_{Jupiter}$ and a orbital inclination of $87.1° \pm 0.2°$. Together with radial velocities measurements of *HD 209458 b* it was possible to derive its mean density and prove that short period gas giant planets were indeed planets.

3.2.2 Instrumental limitations

The radial velocity method continue to prove a reliable method the years following the discovery of Mayor and Queloz (1995), with the first hundred planets discovered around a main-sequence star being made by this method (Perryman, 2011; Mayor, Lovis, and Santos, 2014).

As such to the continue search using this method, it becomes necessary the development of even more precise instruments capable of detecting the signal caused by smaller planets. Fischer et al. (2016) presents a review of the status of the current spectographs used to search for exoplanets with radial velocity measurements. HARPS and HARPS-N spearhead the current observation made in terms of single measure precision, achieving precisions of $0.8ms^{-1}$.

HARPS was commissioned in 2003 on the $3.6m$ telescope at La Silla. Its fiber-fed spectrometer was the first instrument to deliver radial velocity measurements precision better than $1ms^{-1}$. HARPS-N, a copy of HARPS, by its turn was commissioned in 2012 on the Telescopio Nazionale Galileo located at the Roque de Los Muchachos Observatory (Fischer et al., 2016).

They are a cross dispersed echelle spectrograph very similar to UVES at VLT (Pepe, Mayor, and Rupprecht, 2002).The internal uncertainties are obtained calculating the quadratic sum of the photon and readout noise, the wavelength calibration error estimated from the root mean squared (rms) dispersion oh ThAr lines, and instrumental drift error. These errors when combined provide a lower bound to the true error bars of the radial velocity measurements allowing an RV precision of $0.8ms^{-1}$ (Fischer et al., 2016).

Following the successes of HARPS and HARPS-N, the next generation spectrographs take form with ESPRESSO, a modern echelle sprectrograph with extreme radial velocity and spectroscopic precision (Pepe et al., 2013). ESPRESSO is expected to reach precisions of $0.1ms^{-1}$, far superior than its predecessors. With such precision it will be capable of obtaining radial velocity measurements of Earth-like planets in the habitable zone of its stars, as Earth for comparison, creates an RV signal of approximately of $0.09ms^{-1}$.

3.3 Stellar noise contamination

Although astronomers have been trying to build more precise spectrographs capable of detecting Earth-like planets, an important effect to have in account when observing a star is the contamination its activity will cause in the radial velocity measurements obtained. With the commission of HARPS it became possible to reach precisions in the radial velocity measurements below the meter-per-second and the stellar noise signatures, although small, hinders the possibility of detecting Earth-mass planets in its star habitable zone.

3.3.1 Causes

Stellar noise is the result of three different types of perturbations (Dumusque et al., 2011b; Dumusque et al., 2011a). The first noise source comes from the oscillations of solar type stars due to the dilation and contraction of external envelopes. Such oscillations are caused by the propagation in the surface of the star of pressure waves (p-modes) over times scales of minutes. Depending on the spectral type and evolutionary stage of the star, the amplitudes of the p-modes are of the range of tens of centimeters-per-second, but due to the accumulation of the interference of tens of modes can induce noise on the radial velocity measurements in the range of 0.1 to 4 $m\ s^{-1}$.

Another source is due to the convection in the external layers of solar type stars that causes different types of granulation, such as smaller time scales granulation, mesogranulation, and supergranulation. These types of granulation are capable of affecting the radial velocity measurements over time scales of several minutes to hours, and when integrated over the all stellar disk, can induce noise on the range of the meter-per-second.

The last noise source to take into account is due to spots and plages on the surface of the star, that has a usual timescale of tenths of days. The impact of stellar spots on the radial velocity and photometric curves was studied, using the sunspot properties recorded between 1993 and 2003. The radial velocity amplitude varied considerably, between 0.2 and $5ms^{-1}$ for solar-age G stars depending on the activity level, being significantly more quiet and of much smaller amplitudes during the low activity period, and 30 and $50ms^{-1}$ for Hyades-age G stars (Saar

and Donahue, 1997; Santos et al., 2000; Lagrange, Desort, and Meunier, 2010).

Spots are not the only source of noise, and the presence of bright plages also induces a variable signal in radial velocity measurements (Meunier, Desort, and Lagrange, 2010). It was observed that although the noises originated from spots and plages usually compensate each other, it does not compensate entirely due to the surface ratio of spots and plages to vary, and it still becomes necessary to have this source into consideration (Dumusque et al., 2011a).

3.3.2 Contamination examples

Having into consideration that several sources of stellar noise can contaminate our signal, even when having it into consideration it is not always possible to detect them completely. The announcement of the discovery of an exoplanet that followed by a more rigorous analysis questions or disproves its existence has happened several times in the past.

CoRoT-7, for example, is a G9 main sequence star were Léger et al. (2009) discovered the first super-Earth (*CoRoT-7b*) with a measured radius of $1.68 \pm 0.09\ R_{Earth}$ and a period of $0.857509 \pm 3 \times 10^{-5}$ days, through photometric observations with CoRoT[1]. Following these results, and using HARPS, it was not only confirmed the existence of *CoRoT-7b* with the detection of the radial velocity signal of that planet, but also discovered the signal of a second one (*CoRoT-7c*) with a period of 3.69 days and a radial velocity amplitude of $4ms^{-1}$ (Queloz et al., 2009).

In 2010, again analyzing the radial velocity measurements obtained with HARPS, it was not only found the signals of *CoRoT-7b* and *CoRoT-7c* but also evidence for another signal with a period of 9 days possibly from another planet orbiting the star (Hatzes et al., 2010). Haywood et al. (2014) observed once more *CoRoT-7* with HARPS and CoRoT, to this time analyse the star with both radial velocity and photometric data. With it and having into consideration *CoRoT-7* is a more active star than the Sun, it was found that a two planet model plus stellar activity was the most likely scenario for the data obtained. Haywood et al. (2014) was only able to confirm the existence of *CoRoT-7b* and *CoRoT-7c*, and that the 9 days signal was more likely associated with stellar rotation.

[1]Stands for Convection rotation et Transits planétaires, a space telescope in use from 2006 to 2014, with the mission of observing star vibrations and search for exoplanets. Source: `https://corot.cnes.fr/en/COROT/index.htm`

HD41248 is again another example of how stellar noise can influence the data gathered. Jenkins et al. (2013) found two signals in the 62 HARPS archival radial velocities for the star, with periods of 18.357 days and 25.648 days, that seemed to indicate the presence of two planets with a mass of 12.3 and 8.6 M_{Earth}, respectively. Santos et al. (2014), however, after adding 160 new radial velocity points obtained with HARPS, concluded that the 25 days period signal was present in the stellar activity and in the full width half maximum (FWHM) of the HARPS CCF. Given that *HD41248* had a rotation period of approximately 20 days, this let to Santos et al. (2014) to propose that the 18 and 25 days signals observed were most likely caused by the differential rotation pattern of the star.

While this thesis is more focused in the radial velocity method to detect exoplanets, this method is not the only one that is affected by the presence of stellar noise. For example Barros et al. (2013) showed that similar problems arise in photometric time series used for transit detection, as their work concluded that the hot-Jupiter *WASP-10b* was more likely caused by spot occultation features or systematics, instead of a planet with a mass of $0.1 M_{Jupiter}$ previously claimed.

With these examples, it is clear that the search for exoplanets faces several challenges, specially when searching for planets whose radial velocity signals are close to the spectrographs precision and to the stellar activity that will most likely contaminate the measurements with noise. As such the development of tools capable of analyzing the data gathered, having into consideration the stellar events discussed in this chapter is extremely important to future successes. With this in mind, it was the goal in this thesis to analyses and create a tool capable of having these effects into consideration and that is explained in chapter 4.

Chapter 4.
Gedi

"Pass on what you have learned."

Yoda (896 BBY-4 ABY)

With the problem of the stellar noise contamination in radial velocity measurements raised, the need of proper tools to deal with such issue becomes important in order to successful find extra-solar planets. As such, and as part of this thesis work we developed a python packaged named *gedi*[1], to analyze radial velocity measurements with Gaussian processes. The package is available online for free and can be installed by anyone with the command

```
$ pip install Gedi
```

The current version of *gedi* is version 0.2, but is possible that in the near future other versions are released with minor fixes in the package code. The package was developed using Python 2.7.

4.1 Introducing the Gaussian Jedi

There are other tools available to work with Gaussian processes such as *george*[2] (Ambikasaran et al., 2014), and *pyGP*[3] (Neumann et al., 2015). Such tools although widely used and tested, present some challenges such as it becomes difficult to fully read and understand the code and steps taken by them as it performs the calculations. To avoid that, it was decided to

[1] https://github.com/jdavidrcamacho/Gedi

[2] http://dan.iel.fm/george/current/

[3] https://github.com/PMBio/pygp

developed a tool from zero that we fully understood and knew how it worked. An example of the capabilities of *Gedi* is presented in appendix B.

4.1.1 Kernels

Gedi has all the fundamental tools to work with Gaussian processes regression. The most important aspect is the set of kernels that are available. As explained in section 2.1, it was implemented the most basic kernels from which, it can be used to express more complex kernels either by adding or multiplying them.

In *Gedi* the available the kernels are

```
Gedi.kernel.ExpSquared(theta, lenght_scale)
Gedi.kernel.ExpSineSquared(theta, lenght_scale, period)
Gedi.kernel.RatQuadratic(theta, lenght_scale, alpha)
Gedi.kernel.Exponential(theta, lenght_scale)
Gedi.kernel.Matern32(theta, lenght_scale)
Gedi.kernel.Matern52(theta, lenght_scale)
Gedi.kernel.WhiteNoise(theta)
```

that represent the *ES*, *ESS*, *RQ*, *Exp*, *M32*, *M53*, and *WN* kernels, respectively, mentioned in section 2.1. Besides this ones there was also implemented the *quasi-periodic kernel* given by

```
Gedi.kernel.QuasiPeriodic(theta, l1, l2, period)
```

This last kernel could be also given by

```
Gedi.kernel.ExpSineSquared(t1, l1, period) * Gedi.kernel.ExpSquared(t2, l2)
```

but due to its common use it became simpler to have its own kernel implemented. The only difference between the two lines of code come in the representation of the amplitude of the kernel, as it is necessary to have into consideration that

$$theta = t1 \times t2. \tag{4.1}$$

4.1.2 Log marginal likelihood

The most useful aspect to evaluate a given Gaussian process in the log marginal likelihood given in equation 2.26, since it allow us to evaluate what is the best kernel to use in a given dataset. In *gedi*, if we have into consideration an array **t** that represents for example time, an array **y** that represent a set of measurements, and **yerr** that represent the error presented in those measurements. The function

```
Gedi.kernel_likelihood.likelihood(kernel, t, y, yerr)
```

will give back the value of the log marginal likelihood obtained when using a given kernel.

4.1.3 Optimization algorithms

As explained in section 2.2, it is extremely unlikely that the initial hyperparameters given to the kernel are the best ones, and it is then necessary to run a optimization routine to obtained the best possible set of parameters of the kernel. A mechanism to optimize a kernel can be with the use of gradient based algorithms. As presented in section 2.2, *gedi* has three different algorithms implemented, that can be used in two different ways.

The first consists in using a specific algorithm to optimize the kernel. In *gedi* this called `single_optimization`, and can be done with the command

```
Gedi.kernel_optimization.single_optimization(kernel, t, y, yerr, method='BFGS')
```

that will optimize the kernel's parameters using the algorithm mentioned in the `method` parameter. By default *gedi* will use the BFGS algorithm, but in can use the steepest descent algorithm (`SDA`) or the alternative steepest descent algorithm (`altSDA`).

The second form of using gradient based algorithms in *gedi* is called `committed_optimization`, and arise from the problem that different algorithms will perform better in different sets of data. As such the `committed_optimization` uses the three available algorithms and in the end returns the result of the one that performed better.

Similarly to the command for the `single_optimization`, this type of optimization is used with the command

```
Gedi.kernel_optimization.committed_optimization(kernel, t, y, yerr)
```

Besides the final log marginal likelihood and the final hyperparameters, it is also possible to know which was the best algorithm adding the parameter `return_method=True` to the `committed_optimization`.

As in the case of *george*, it is possible to use `scipy.optimize` algorithms with *gedi*. This in its turn makes possible the use other optimization algorithms that are not implemented in *Gedi*, such as the *conjugate gradient algorithm* (Nocedal and Wright, 2006), to find the best parameters to a given kernel. The full list of algorithms available in `scipy.optimize` can be consulted on-line [4].

4.1.4 Markov Chain Monte Carlo

It is possible to also use MCMC in *gedi* to optimize our kernels. *Gedi* has a simple random walk Metropolis-Hastings algorithm implemented that can be used with the command

```
Gedi.kernel_mcmc.MCMC(kernel, t, y, yerr, hyperparameters, runs, burns)
```

where, as previously, **kernel**, **t**, **y**, and **yerr**, represent the kernel being used, time, the array of measurements, and the error in those measurements, respectively. The new parameter **hyperparameters** is an array with the kernel's hyperparameters, **runs** will set the number of iterations our Metropolis-Hastings algorithm will take, and **burns** represent the number of iterations that will be used as burn-in in the beginning of the MCMC.

Besides this it is also possible to use the Python package *emcee*[5] developed by Foreman-Mackey et al. (2013) that implements the Affine Invariant Markov chain Monte Carlo Ensemble sampler together with *gedi* (Goodman and Weare, 2010). In appendix B an example will be shown on how this two packages can be combined and what results can we obtained with them. Since *emcee* proved to be a more efficient and fast tool than the MCMC implemented in *gedi*, it will be used in chapter 5.

[4] `https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.optimize.minimize.html`

[5] `http://dan.iel.fm/emcee/current/`

4.2 Tests and performance

George is probably one of the most used python packages to work with Gaussian processes, and as such it is impossible not to ask how this package and *gedi* compare to each other. To check how *gedi* performed when compared with *george* we ran several test in a Intel® Core™ i3-350M running at 2.27Ghz and 3.7 GB of RAM.

4.2.1 Covariance matrix calculation

The most important property of a kernel is its covariance matrix K. This matrix will be use in the calculation of the log marginal likelihood, necessary to analyze different models, and it is going to be used almost much all calculations necessary when we are trying to perform Gaussian process regression. As such it is important to a inquire how much time the construction of this matrix takes, as it will influence the total time required to obtain a final result.

For that we can compare the time taken to build the covariance matrix, given the same initial conditions, that is, using the exact same kernels and datasets, for different types of kernels, and how it performs dealing with the sum and product of kernel, since in most real word problem we will most likely use a combination of these operations.

For this we simulated sine-type data with noise similar to the one in appendix B and that is available on Github[6]. Afterward it was used different kernels to built the respective covariance matrix, and evaluated the performance in data sets of different size, containing for 10 to 500 points.

The first test, shown in figure 4.1, allow us to see the performance of *gedi* building the covariance matrix of the *ES kernel*, *ESS kernel*, and *RQ kernel* with their existing versions implemented in *george*. *Gedi* perform slightly better for fewer points, but the difference disappear as the number of points increases. On the long run in seems that *gedi* shows as good of a performance as *george*.

A similar behavior was observed in the analysis of the sum of two kernels. In figure 4.2 is seen that although *george*, once more builds the covariance matrix slower for smaller matrices,

[6] https://github.com/jdavidrcamacho/Tests_GP/tree/master/MSc_results

as the number of points increases, the difference between the two packages disappear and no significant difference is seen for large values of n.

The last test made in the covariance matrix construction, was performed to the *quasi-periodic kernel*. Unlike *gedi*, *george* does not have this kernel implemented since it can be done with the multiplication of the *ESS kernel* and the *ES kernel*. As such and for a better analysis, it was compared the performance of the product of this two kernels with *gedi* and *george* and the *quasi-periodic kernel* implemented in *gedi*.

As seen in figure 4.3, while the product of kernels have a performance for both *gedi* and *george* identical to the previous tests, the covariance matrix calculation of *gedi*'s *quasi-periodic kernel* seem to, perform slightly better for larger matrices. With these results, we can say with confidence that *gedi* performs as well or better than *george* in the calculation the covariance matrix.

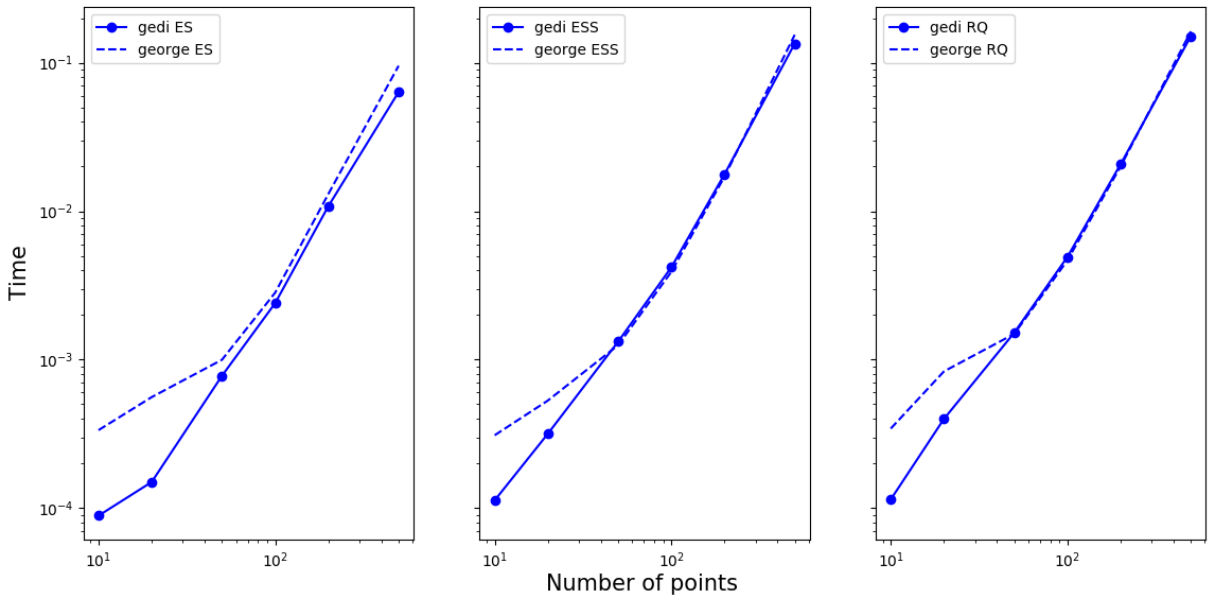


Figure 4.1: Comparison of the three covariance matrix calculations. On the left using *gedi*' squared exponential kernel (gedi ES) and *george*'s version (george ES), in the middle using *gedi*' periodic kernel (gedi ESS) and respective *george*'s version (george ESS), and on the right *gedi*' rational quadratic kernel (gedi RQ) and *george*'s version (george RQ).

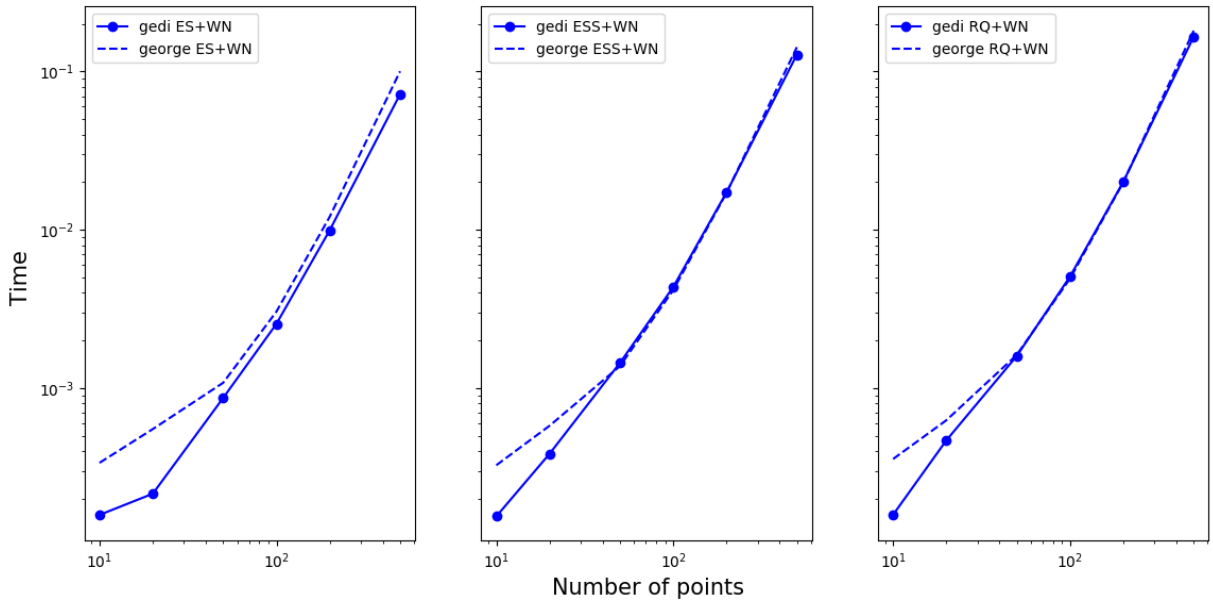


Figure 4.2: Comparison of the covariance matrix calculations. From left to right, *gedi*'s squared exponential kernel plus a white noise kernel (gedi ES+WN), a periodic kernel plus a white noise kernel (gedi ESS+WN), and the rational quadratic kernel plus white noise (gedi RQ+WN). Each compared with its *george*'s version.

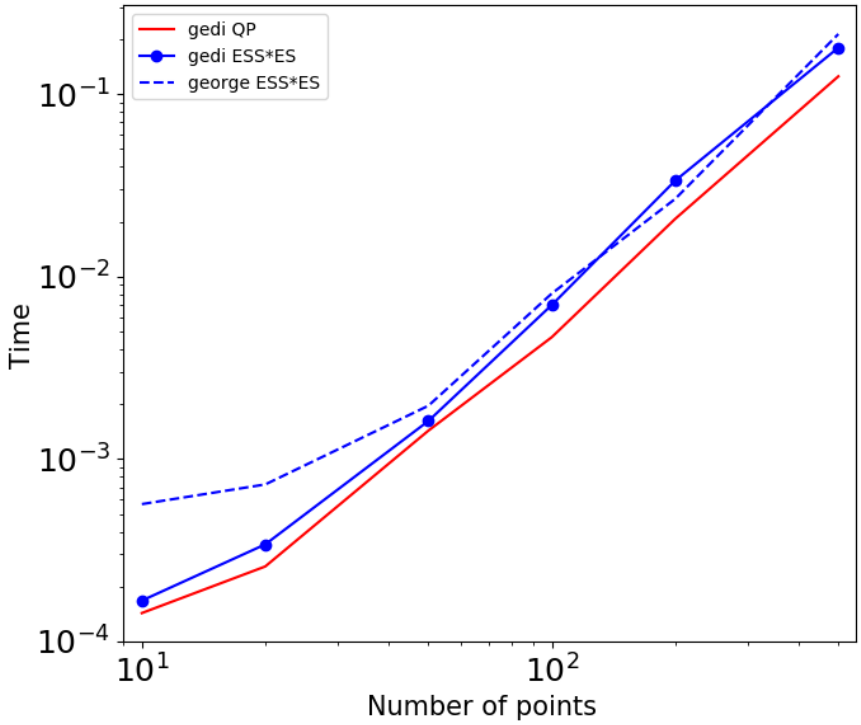


Figure 4.3: Comparison of the covariance matrix calculations done using *gedi*'s quasi-periodic kernel (gedi QP), the product of *gedi*'s periodic kernel with the squared exponential kernel (gedi ESS*ES) and *george*'s version (george ESS*ES).

4.2.2 Optimization

Another test between *gedi* and *george* can be made with the total time spent in optimizing the kernel's hyperparameters. Since both packages are able to use `scipy.optimize` to perform this, we can use it to perform a comparison in the speed of *scipy.optimize* while using the two packages.

For it is only required to *gedi* and *george* to be capable of calculating the log marginal likelihood and the gradient of the kernels, since *scipy.optimize* will use gradient based algorithms to find the best hyperparameters. As such, it will be the efficiency of both *gedi* and *george* in this two operations that will influence the final performance in the optimization on a given kernel.

As in the previous test, simulated sine-type datasets were used from 10 to 500 points, and *scipy.optimize* ran until an optimal value for the different hyperparameters. As mentioned in the previous test all programs used for this test are available on-line.

When doing the optimization of a set of kernel, it is important to be careful with the initial parameters given to such kernels, as the algorithms might get suck in a solution that is not the optimal solution. In these tests we analyzed more carefully the time spent by each package, and was observed that they both gave similar solutions and likelihoods. However, we do not check if those are the "true" optimal values to the given datasets.

As in the covariance matrix calculation, for this section first test, it were used the same kernels. In figure 4.4 it is possible to see that *george* had a fairly better performance independently as the number of points increased for the *ESS* and *RQ kernels*, while *gedi* and *george* show similar results for the *ES kernel*.

The second test in this section, was done adding a *WN kernel* to the previously used kernels to observe how the sum of two kernels would behave. It can be seen in figure 4.5 and shows that *george*'s performance surpasses *gedi*'s performance in all datasets this time, and as the number of points increased the gap between the two packages also increases.

Such disparities while performing the optimization of the kernel's hyperparameters can be explained due to the fact that `scipy.optimize` algorithms are gradient based algorithms. Due to that it is necessary to have into consideration that when the derivative of the kernels are used

by `scipy.optimize`, they are written in C++ in *george*, while *gedi* has it written in python.

As such is fair to assume that *george* will always outperform *gedi* if gradient based algorithms are used to optimize the kernels.

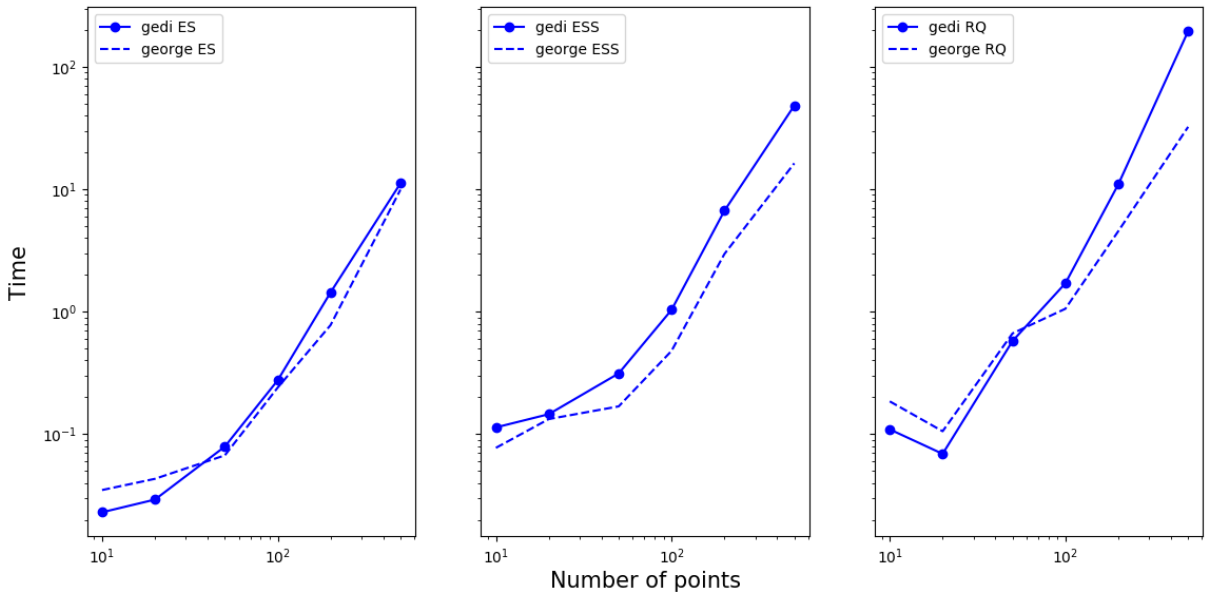


Figure 4.4: Comparison of the time taken by `scipy.optimize` in the optimization of the *gedi*' squared exponential kernel (gedi ES), periodic kernel (gedi ESS), and the rational quadratic kernel (gedi RQ) with the respective *george*'s versions.

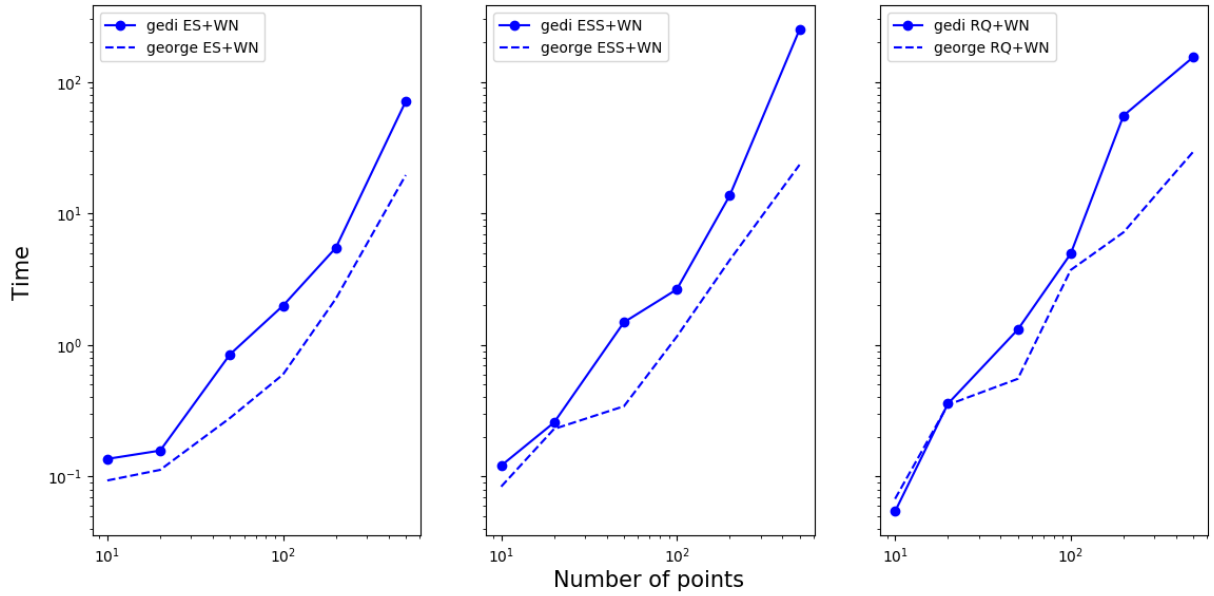


Figure 4.5: Comparison of the time taken by `scipy.optimize` in the optimization of the sum of *gedi*' squared exponential kernel, periodic kernel, and rational quadratic kernel with a white noise kernel (gedi ES+WN, gedi ESS+WN, and gedi RQ+WN respectively). As previously they are all compared with *george*'s versions.

4.2.3 Markov Chain Monte Carlo

The last analysis in performance can be made with the use of *emcee* package, and compare how both packages perform when a MCMC is used with them. Like the previous tests, it was used the simulated sine-type data mentioned earlier, and the same data sets used in the tests with *scipy.optimize*.

Similarly to the previous subsection it was given a more closer look at the times spent using the MCMC with different types of kernels and different number of points. While in a MCMC it is important to have achieved convergence in the parameters, the tests done here were limited to 1000 burn-in's followed by another 2000 steps, as such convergence is not guaranteed. In chapter 5 it will be given a much careful analysis of the data that was worked there.

The first tests used, once again, a single *ES kernel*, *ESS kernel* and *RQ kernel*. From figure 4.6 it is possible to observe that unlike what happened with *scipy.optimize*, *gedi* manages to be considerable faster when used together with *emcee* than *george* for every data set analyzed.

A similar behavior is once again observed in figure 4.7, were it was analyzed the sum of a *ES kernel*, *ESS kernel*, and a *RQ kernel* with a *WN kernel*.

With this simple tests it seems that *gedi*, given the exact same initial conditions, appears to be considerable faster when used with MCMC than *george*. Such behavior might occur due to *gedi* works in a simpler manner. While *gedi* simply calculates the necessary log marginal likelihood necessary for the MCMC, *george* first creates a `gp.object` in order to calculate the log marginal likelihood, such object is made to allow *george* to perform other operations such as prediction with Gaussian processes if the user wants it, but not necessary for the use of an MCMC in our analysis. Thus it is our believe that the extra calculations increases the time taken by *emcee* while using this package.

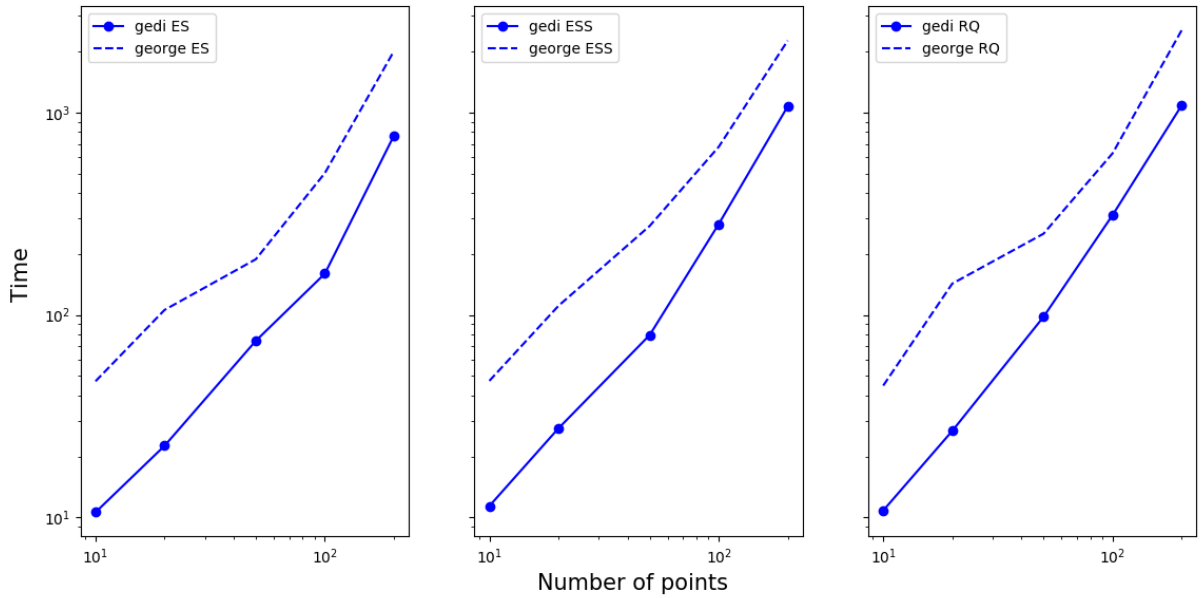


Figure 4.6: Comparison of the time taken by `emcee` running an MCMC with 1000 burn-ins and 2000 steps of the *gedi*' squared exponential kernel (gedi ES), periodic kernel (gedi ESS), and rational quadratic kernel (RQ kernel). All three kernels are compared with the respective *george*'s version.

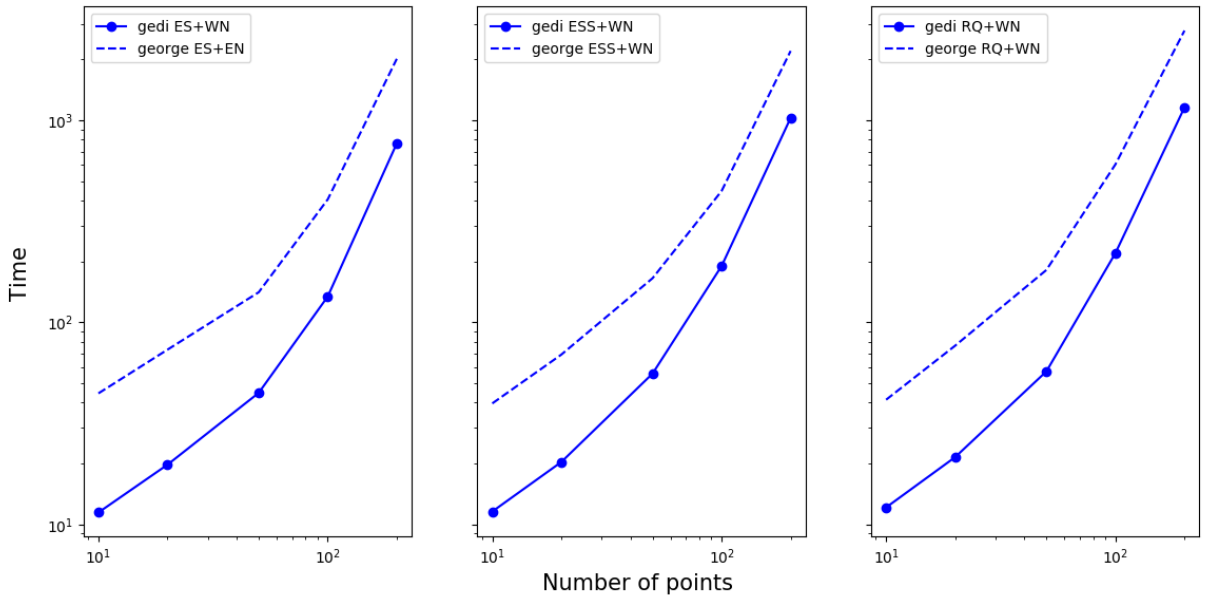


Figure 4.7: Comparison of the time taken by `emcee` running an MCMC with 1000 burn-ins and 2000 steps of the sum of *gedi*' exponential squared, sine squared exponential, and rational quadratic kernels with white noise kernel (gedi ES+WN, gedi ESS+WN, and gedi RQ+WN respectively) and *george*'s version of such sums.

Chapter 5.

Results

"We must trust to nothing but facts, these are presented to us by Nature, and cannot deceive."

Antoine Lavoisier (1743-1794)

After showing how the package developed in this thesis works, the next step will be using simulated data to analyze the effects of starspots in radial velocity measurements. Using such data with *gedi*, the goal is to determine the accuracy of the package into obtaining the rotation period of the star, and if it is possible to obtain the orbital parameters of a planet in radial velocity measurements contaminated with an activity signal.

We will use use the freely available SOAP 2.0[1] tool, which is capable of estimating photometric and radial velocity variations induced by active regions on the surface of stars (Dumusque, Boisse, and Santos, 2014). As a consequence of this, SOAP 2.0 is a useful tool to simulate spots in a solar-type star whose properties, such the as rotational period of the star, where known a priori.

All analysis done in the next section will use *gedi* and *emcee* to determine the best results possible with the use of an MCMC, and then calculate the median of our values with the 16th and 84th percentile values corresponding to the limits of our confidence interval, although it has been shown that the mean tends to be more stable than the median and the mode (Mcleod and Quenneville, 2001; Hamra, MacLehose, and Richardson, 2013).

[1] `http://www.astro.up.pt/resources/soap2/`

5.1 Data analysis

For the following tests we consider four different datasets containing the RV variations caused by the presence 1, 5, 10, and 20 spots in the surface of a solar-type star without any planet orbiting it. Using SOAP 2.0, we randomly generated spots on the surface of a solar-type star, with a size up to 0.3 R_{Sun} (Sun's radius $\sim 7 \times 10^5$ km), in a latitude interval between 35 and 15 degrees, and randomly distributed on both hemispheres. Afterwards we also included randomly generated white noise between the 0.2 and 0.5 m/s, a level of precision close to the one expected of ESPRESSO.

These datasets should allow for the detection of the stellar rotation period of 25.05 days, and to disentangle this rotation signal from possible planets. We used two different kernels capable of extracting physical information from the data. The periodic and the quasi-periodic kernels contain hyperparameters that relate the periodicities in the data, and as such make the perfect candidates to successful detect the stellar rotation period.

Using these two kernels, and an additional white noise kernel, and for each dataset, we made six different analysis. The first one consisted in a dataset that contained one measurement every day during 100 days, while in the second analysis the same measurements where multiplied by a linear function in order to simulate a linear decay in the spots signal, to simulate spot evolution. For the third analysis, the equivalent of 30 days of measurements were removed from the data of the second analysis in order to simulate a possible lack of measurements common in real observations. For the forth, fifth, and sixth analysis the datasets contained a measurement every four days during the same period of 100 days. The fifth analysis contained a linear decay similar to the second analysis, while in the sixth 30 days of measurements were removed to simulate absence of measurements.

5.1.1 One spot analysis

The first analysis consisted in a dataset that contained the radial velocity signal induced by only one starspot. Combining *gedi* and *emcee*, we made six different analysis of this data set for each of the two kernels. Tables 5.1, 5.2, 5.3, and 5.4 lists our results.

For the periodic kernel, when having one measurement per day, the period converged to values around 25 days, close to the real value, but never reaching the 25.05 days period even when considering the errors margins. The estimate for the stellar rotation period has a relative error of around 0.2% in the three cases.

Table 5.2 shows the results for the quasi-periodic kernel. With this kernel, the period hyper-parameter also converged to a value close to the real rotation period of the star. Both analyses, with the unaltered data and with a linear decay obtain again a relative error around 0.2%, and the addition of a gap slightly increased it to around 1.7%.

Another important observation is that, unlike the periodic kernel, the quasi-periodic kernel had a white noise parameter converged to values close to zero on the three cases. In the periodic kernel although the white noise of the first analysis is smaller than expected, when a linear decay and a linear decay plus a gap were added to the data, the white noise parameter obtained was an order of magnitude larger than expected. Since the linear decay considerably changed the amplitude of the signal, it is possible that this decay is being classified as created by white noise by the periodic kernel. In the quasi-periodic case the white noise seems to be absorbed either by the amplitude or the aperiodic length-scale (length-scale 2) and thus converging to a small value around 0.002 m/s.

This of course will have an impact in the amplitudes estimated by both kernels. In the periodic kernel the decrease in the amplitude in each analysis is compensated by an increase in the white noise amplitude, but the estimated amplitudes, having into consideration the error intervals, are similar. The quasi-periodic kernel on the other hand, obtained a different estimation for the amplitude for each case but still within the uncertainties of one another.

One last, and expected, characteristic of the estimated parameters in the quasi-periodic kernel is seen in the length-scale 2 when using data without decay. As the data is clearly periodic it would be expected that the aperiodic length-scale would converge to a high value, as it would

Table 5.1: MCMC results for the different analysis of a one starspot signal with the periodic kernel while having a measurement per day. The first analysis (Normal) contained the data obtained from SOAP without any alterations besides the addition of noise. The second (Decay) contained the previous data multiplied by a linear function to simulate a simplified form of spot evolution. In the third analysis (Decay and gap), it was created a 30 days gap in the data to simulate a lack of observations.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$23.356^{+8.517}_{-5.491}$	$19.981^{+11.809}_{-5.982}$	$19.565^{+12.371}_{-6.010}$
Length-scale	$0.754^{+0.086}_{-0.072}$	$0.839^{+0.064}_{-0.056}$	$0.828^{+0.238}_{-0.165}$
Period (days)	$25.003^{+0.002}_{-0.002}$	$25.009^{+0.040}_{-0.041}$	$25.009^{+0.048}_{-0.048}$
White noise (m/s)	$0.039^{+0.057}_{-0.039}$	$2.894^{+0.233}_{-0.215}$	$3.436^{+0.347}_{-0.298}$

Table 5.2: MCMC results for the analysis of one starspot RV signal using a quasi-periodic kernel, and having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$20.370^{+6.650}_{-4.223}$	$27.562^{+11.702}_{-6.894}$	$23.601^{+9.044}_{-5.609}$
Length-scale 1	$0.727^{+0.064}_{-0.056}$	$0.810^{+0.077}_{-0.065}$	$0.792^{+0.074}_{-0.068}$
Length-scale 2 (days)	$13057.775^{+6161.550}_{-5036.670}$	$300.698^{+87.952}_{-65.181}$	$242.168^{+67.325}_{-54.577}$
Period (days)	$24.994^{+0.003}_{-0.003}$	$25.001^{+0.025}_{-0.027}$	$24.985^{+0.032}_{-0.033}$
White noise (m/s)	$0.002^{+0.017}_{-0.002}$	$0.002^{+0.018}_{-0.002}$	$0.002^{+0.002}_{-0.002}$

make the sine term in the quasi-periodic kernel (equation 2.16) dominate.

After obtaining the estimated hyperparameters for each kernel for all the three different conditions we imposed, it is necessary to compare which one seemed to fit better the given data. For this we can look at the log marginal likelihood of each kernel, as stated in chapter 2.2.2. The MCMC allows us to obtain a log marginal likelihood histogram for each analysis, shown in figure 5.1.

We can observe that there is a significant difference between the two kernels in the datasets with a linear decay and a linear decay plus a gap. The quasi-periodic kernel (red histograms of figure 5.1) obtain a better result, that is a higher log marginal likelihood, in comparison with the periodic kernel (blue histograms of figure 5.1). In the analysis of the dataset without decay and/or gaps, the periodic kernel performed slightly better, but as was expected the quasi-periodic kernel is having a similar behavior to the periodic kernel as its length-scale 2 tends to very high values, making the quasi-periodic kernel to behave similarly to he periodic kernel.

When substituting the datasets by the ones containing a measurement every four days similar

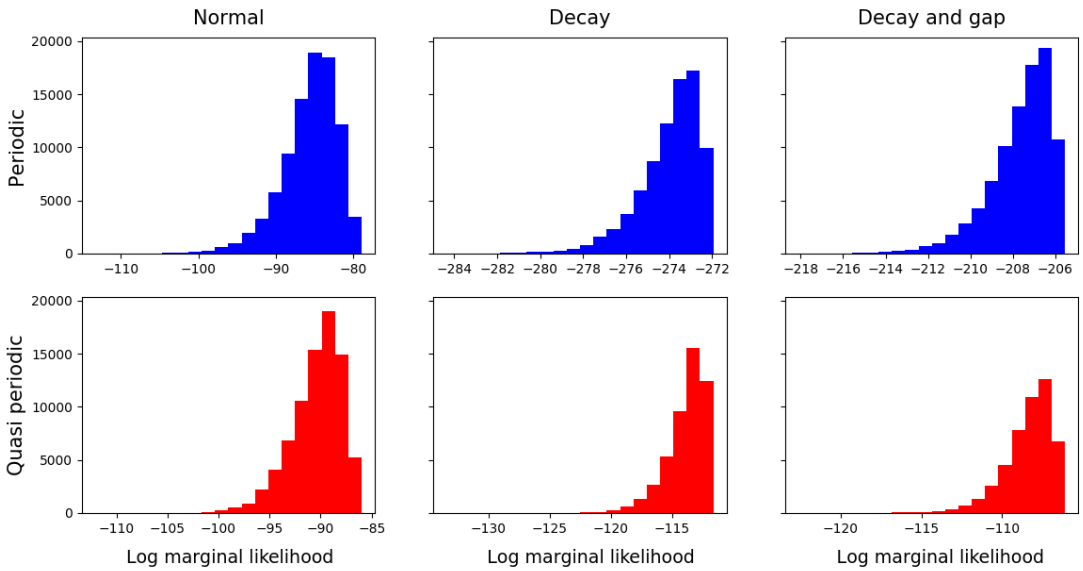


Figure 5.1: Log marginal likelihood obtained for each kernel while analyzing the RV signal of one spot with a measurement per day.

results are observed. Both kernels seem to have the period parameter converging to a value close the real value of 25.05 days but again smaller than this value.

When using a periodic kernel the period estimated for the SOAP data plus noise had a relative error of around 0.3%. The data with a linear decay obtained a higher relative error of about 3.7%, but when a gap was created the relative error decreased to about 0.1%.

As in the previous estimations, the white noise parameter of the kernel is absorbed in the data without decay but even more with the data with decay. In the data with decay and gap, we again seem to see the decay in the data being absorbed by the white noise term that reached values of an order of magnitude higher than expected.

In the quasi periodic kernel the relative errors obtained for the period were around 0.2%, 0.8%, and 0.6% for the data without decay, the one with decay, and the one with decay and gap, respectively. Like in the estimated parameters using the datasets with a measurement per day, the white noise term converged to a smaller value than what was expected, and a similar interpretation can be made as the one done previously analysis done for the quasi periodic kernel estimations of table 5.2.

After observing this new estimations, we need again to look at the log marginal likelihood obtained for each kernel, to determine what kernel better fits this datasets of a measurement every four days. In figure 5.2 we observe that the quasi-periodic kernel obtains a slightly better

Table 5.3: MCMC results for the analysis for a one spot radial velocity signal with the periodic kernel, while having a measurement every four days. As previously it was used a dataset containing the SOAP generated data with noise (Normal), a dataset were the data was multiply by a linear decay (Decay), and a dataset were a 30 days gap was created (Decay and gap).

	Normal	Decay	Decay and gap
Amplitude (m/s)	$29.367^{+16.655}_{-8.782}$	$15.648^{+3.589}_{-2.778}$	$16.177^{+7.523}_{-3.987}$
Length-scale	$0.885^{+0.163}_{-0.126}$	$0.225^{+0.116}_{-0.164}$	$0.558^{+0.220}_{-0.206}$
Period (days)	$24.981^{+0.013}_{-0.014}$	$24.128^{+0.392}_{-0.437}$	$25.028^{+0.120}_{-0.121}$
White noise (m/s)	$0.011^{+0.175}_{-0.010}$	$0.004^{+0.227}_{-0.004}$	$3.809^{+1.395}_{-0.990}$

Table 5.4: MCMC results for the analysis for a one spot signal using the quasi-periodic kernel, with a measurement every four days, and having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$22.085^{+9.484}_{-5.118}$	$17.879^{+7.002}_{-3.763}$	$17.035^{+6.288}_{-3.868}$
Length-scale 1	$0.721^{+0.108}_{-0.084}$	$0.720^{+0.128}_{-0.137}$	$0.594^{+0.157}_{-0.131}$
Length-scale 2 (days)	$6766.889^{+8279.036}_{-4014.836}$	$142.438^{+52.160}_{-34.826}$	$136.046^{+72.852}_{-40.390}$
Period (days)	$25.008^{+0.013}_{-0.013}$	$24.852^{+0.103}_{-0.150}$	$24.912^{+0.172}_{-0.200}$
White noise (m/s)	$0.016^{+0.194}_{-0.015}$	$0.008^{+0.121}_{-0.008}$	$0.012^{+0.658}_{-0.012}$

result for the three analysis, although the difference between the two kernel are smaller that in the datasets with one measurement per day.

With the periods obtained and its error margins, we can state that when having just one starspot, in all cases including when spot evolution is simulated, the quasi-periodic kernel fits better the data, but both kernels are capable of determining a close value to the rotation period of the star.

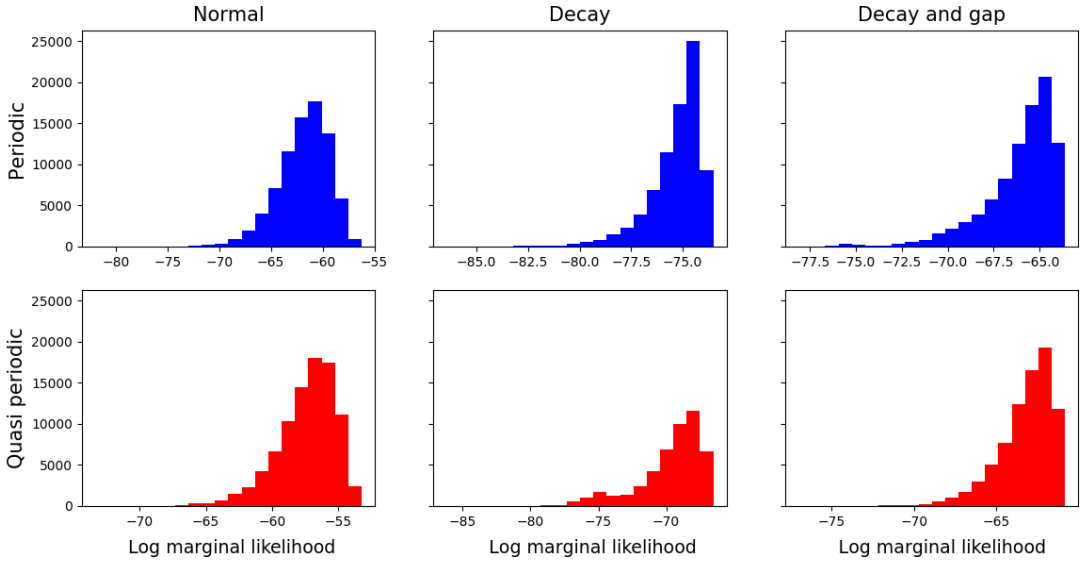


Figure 5.2: Log marginal likelihood obtained for each kernel while analyzing the RV signal of one spot with a measurement every 4 days.

5.1.2 Five spots analysis

The second group of datasets analyzed contained the radial velocity signal of five spots, again created using SOAP. Like in the analysis of just one spot, the original dataset was edited into six different datasets to be used by the periodic and the quasi-periodical kernels.

From table 5.5 we can see the estimated parameters of the periodic kernel when using the datasets with the SOAP data plus noise (Normal), the SOAP data plus noise and a linear decay (Decay), and SOAP data plus noise, a linear decay, and a 30 days gap (Decay and gap), all of which had a measurement per day. Again it is seen that the period, although converging to a value close to the expected 25.05 days, never reaches it. In all three analysis the periods estimated had a relative error of around 0.2%, similar to what was observed in the periodic kernel results of table 5.1. Another characteristic that was already observed is the fact that when decay and decay with a gap is added to the data, the periodic kernel seem to consider this as white noise, as an increase in the white noise estimation is observed for these two cases.

Table 5.6 shows that the quasi-periodic kernel has a similar behavior to the one observed when studying the case of just one spot with measurements every day. The three cases had an estimated period that seemed to converge to the real value but never reaching it, and getting values with a relative error of around 0.2%. As it was observed in the previous analysis of this

Table 5.5: MCMC results for the analysis of five spots with the periodic kernel having a dataset with measurement per day, with a duration of 100 days, and having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$53.314^{+14.873}_{-10.466}$	$73.763^{+57.189}_{-25.822}$	$65.630^{+45.710}_{-21.338}$
Length-scale	$0.652^{+0.039}_{-0.039}$	$1.008^{+0.286}_{-0.200}$	$0.955^{+0.283}_{-0.192}$
Period (days)	$25.000^{+0.001}_{-0.001}$	$24.999^{+0.041}_{-0.042}$	$25.000^{+0.047}_{-0.048}$
White noise (m/s)	$0.001^{+0.010}_{-0.001}$	$8.619^{+0.640}_{-0.596}$	$9.424^{+0.403}_{-0.574}$

Table 5.6: MCMC results for the analysis of five spots with the quasi-periodic kernel having a dataset with measurement per day, with a duration of 100 days, and having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$45.124^{+9.946}_{-7.530}$	$102.970^{+42.969}_{-26.380}$	$78.399^{+31.112}_{-20.088}$
Length-scale 1	$0.645^{+0.039}_{-0.042}$	$0.892^{+0.083}_{-0.078}$	$0.840^{+0.083}_{-0.071}$
Length-scale 2 (days)	$18121.232^{+2753.450}_{-4317.833}$	$387.987^{+120.428}_{-84.783}$	$299.995^{+97.889}_{-63.593}$
Period (days)	$25.001^{+0.001}_{-0.001}$	$25.015^{+0.018}_{-0.016}$	$24.989^{+0.025}_{-0.025}$
White noise (m/s)	$0.009^{+0.057}_{-0.009}$	$0.002^{+0.020}_{-0.002}$	$0.004^{+0.063}_{-0.004}$

kernel, the aperiodic length-scale tend to a high value for the case were the data without decay, as expected.

Once again, if we look at the log marginal likelihood distribution for the six cases, the quasi-periodic kernel shows to be a better choice, specially in the datasets with a linear decay, and a linear decay with a 30 days gap of missing data, as it is capable of obtaining a significantly higher value for the log marginal likelihood.

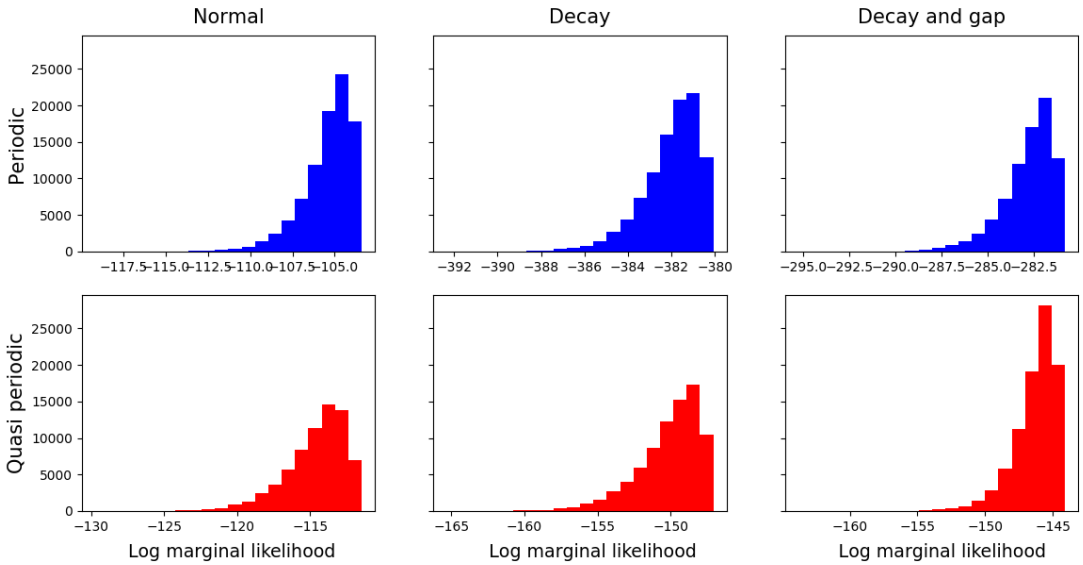


Figure 5.3: Log marginal likelihood obtained for each kernel in the analysis of the radial velocity signals generated by five spots while having dataset with a measurement per day.

As done previously we now consider datasets containing one measurement every four days, and new analysis were made using both kernels. Again we use a dataset that contained the SOAP data plus noise, another dataset were the data was multiplied by a linear decay, and a last one containing data with a linear decay and a gap to simulate lack of observations.

Having into consideration that the data used this time contained five stops, and a different amplitude of the signal, the results for the periodic kernel can have a similar interpretation to the ones done for table 5.3. Looking at the estimated period, all cases had periods converging to values close to the real one.

An interesting characteristic not mentioned earlier, which is also observed in table 5.3, is that when estimating the parameters for the dataset with the linear decay the length scale decreased significantly in comparison with the two other datasets as seen in 5.7. This unexpected result might explain the reason for the estimated period having this case a relative error around the 4.3% while the other cases have errors between 0.1% and 0.2%, an order of magnitude lower.

Table 5.8 presents us the results obtained for the quasi periodic kernel, with similar conclusions to that of table 5.4. The period estimated once was close to the real value, with relative errors between 0.2% and 0.4%, as small as previously obtained. In this case only the analysis of the dataset with a linear decay and a gap obtained an error margin that includes the real value.

Table 5.7: MCMC results for the analysis of five spots with the periodic kernel while having a measurement every 4 days, having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$98.613^{+55.671}_{-30.698}$	$49.291^{+13.089}_{-8.422}$	$45.870^{+21.722}_{-10.820}$
Length-scale	$1.025^{+0.161}_{-0.138}$	$0.171^{+0.093}_{-0.112}$	$0.604^{+0.207}_{-0.198}$
Period (days)	$25.001^{+0.002}_{-0.002}$	$23.971^{+0.322}_{-0.305}$	$25.078^{+0.133}_{-0.120}$
White noise (m/s)	$0.004^{+0.060}_{-0.004}$	$0.006^{+0.189}_{-0.006}$	$8.243^{+1.190}_{-1.696}$

Table 5.8: MCMC results for the analysis of five spots with the quasi-periodic kernel while having a measurement every 4 days, having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$70.499^{+25.657}_{-18.829}$	$67.049^{+27.811}_{-18.462}$	$51.018^{+21.665}_{-12.079}$
Length-scale 1	$0.814^{+0.123}_{-0.088}$	$0.883^{+0.153}_{-0.129}$	$0.699^{+0.163}_{-0.145}$
Length-scale 2 (days)	$7735.555^{+8425.395}_{-4158.568}$	$185.198^{+59.689}_{-42.262}$	$150.954^{+63.298}_{-43.130}$
Period (days)	$24.998^{+0.007}_{-0.006}$	$24.960^{+0.069}_{-0.081}$	$24.981^{+0.139}_{-0.156}$
White noise (m/s)	$0.044^{+0.362}_{-0.043}$	$0.006^{+0.162}_{-0.006}$	$0.013^{+0.990}_{-0.013}$

If we look again at the log marginal likelihood distributions (figure 5.4), the quasi-periodic kernel obtains once more slightly better results. A result that is expected since the quasi-periodic kernel should be better prepared to fit data that is not perfectly periodic.

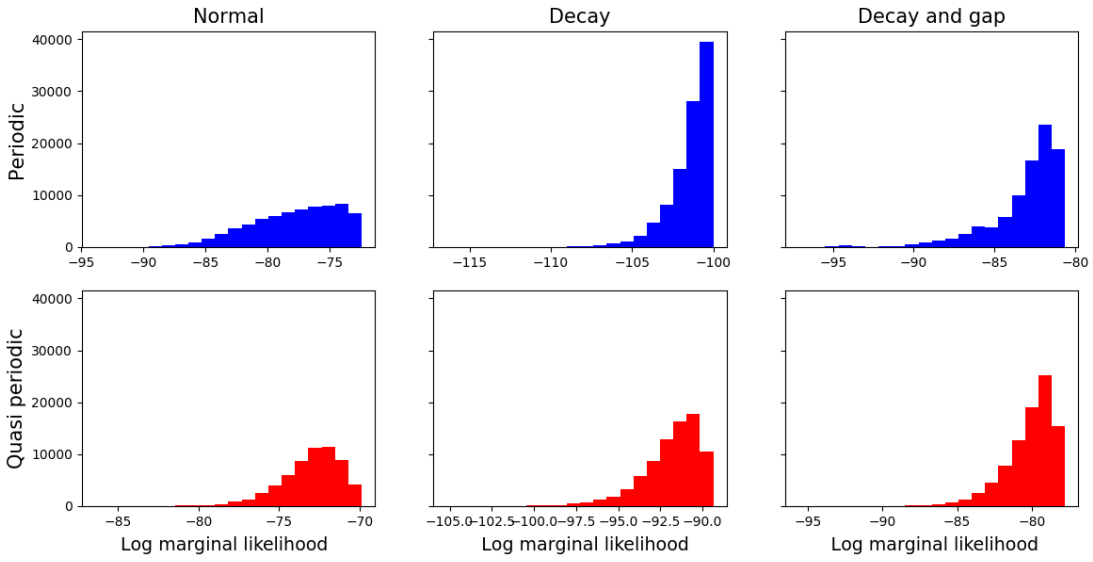


Figure 5.4: Log marginal likelihood obtained for each kernel while analyzing the RV signal of five spots with a measurement every 4 days.

5.1.3 Ten spots analysis

The third round of analysis contained a dataset containing the signal of ten spots, once again generated using SOAP. Like the previous two sections for the periodic and the quasi-periodic kernel the original dataset was edit to obtain one with just the SOAP data plus noise, another with a linear decay in the radial velocity signal, and one with a linear decay and a gap in the data. For each version we consider one measurement per day, and one measurement every four days.

Looking at the results for the periodic kernel for a measurement per day during 100 days, in table 5.9, we can once again see that the period converge to a value close to the expected, having a relative error of around 0.2% in the three cases. The amplitudes obtained show that, when using the dataset with a linear decay and a linear decay with a gap tend to decrease while the white noise amplitude tend to increase to value a order of magnitude higher than expected. This shows, as mentioned earlier, that the periodic kernel seem to interpret the existence of a linear decay and a gap as noise in the data.

For the same analysis made with the quasi-periodic kernel (table 5.10), the estimated periods again converge to a value close to the real 25.05 days, with relative errors ranging from 0.1% to 0.3%. As expected the dataset with the SOAP data without decay had a length-scale 2 or aperiodic length-scale tending to a very high value due to the data being periodic, while the other two datasets had estimated values that were two orders of magnitude lower in comparison. Another behavior already observed in the analysis of the data with one and five spots, comes from the white noise. Once again all three estimations show the white noise converging to a value close to 0, instead of a value between 0.2 and 0.5 m/s, that would correspond to the noise added to the original SOAP data.

Having obtained the estimations for both kernels, we again resorted to the log marginal likelihood to compared the kernels. From the histograms of figure 5.5 made with the results of the MCMC we can see that the periodic kernel seems to fit better when we have data that is clearly periodic, while when the datasets with decay and decay and gap are analyzed the quasi-periodic kernel shows, once more, to be the better kernel.

Table 5.9: MCMC results for the analysis of ten spots with the periodic kernel while having a measurement per day, and having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$46.024^{+12.382}_{-8.993}$	$60.174^{+43.963}_{-20.330}$	$54.787^{+41.622}_{-18.003}$
Length-scale	$0.691^{+0.049}_{-0.042}$	$1.101^{+0.314}_{-0.227}$	$1.052^{+0.348}_{-0.225}$
Period (days)	$25.001^{+0.001}_{-0.001}$	$24.995^{+0.057}_{-0.056}$	$24.997^{+0.068}_{-0.065}$
White noise (m/s)	$0.003^{+0.039}_{-0.003}$	$8.330^{+0.667}_{-0.572}$	$9.917^{+0.986}_{-0.830}$

Table 5.10: MCMC results for the analysis of ten spots with the quasi-periodic kernel while having a measurement per day, and having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$39.804^{+8.855}_{-6.410}$	$60.909^{+25.913}_{-14.311}$	$78.459^{+33.220}_{-20.988}$
Length-scale 1	$0.701^{+0.045}_{-0.044}$	$0.826^{+0.074}_{-0.061}$	$0.930^{+0.074}_{-0.067}$
Length-scale 2 (days)	$16808.872^{+3768.800}_{-4913.686}$	$317.714^{+111.537}_{-69.443}$	$364.195^{+124.136}_{-93.102}$
Period (days)	$25.001^{+0.002}_{-0.002}$	$25.021^{+0.027}_{-0.028}$	$24.984^{+0.028}_{-0.032}$
White noise (m/s)	$0.001^{+0.010}_{-0.001}$	$0.013^{+0.078}_{-0.013}$	$0.002^{+0.028}_{-0.002}$

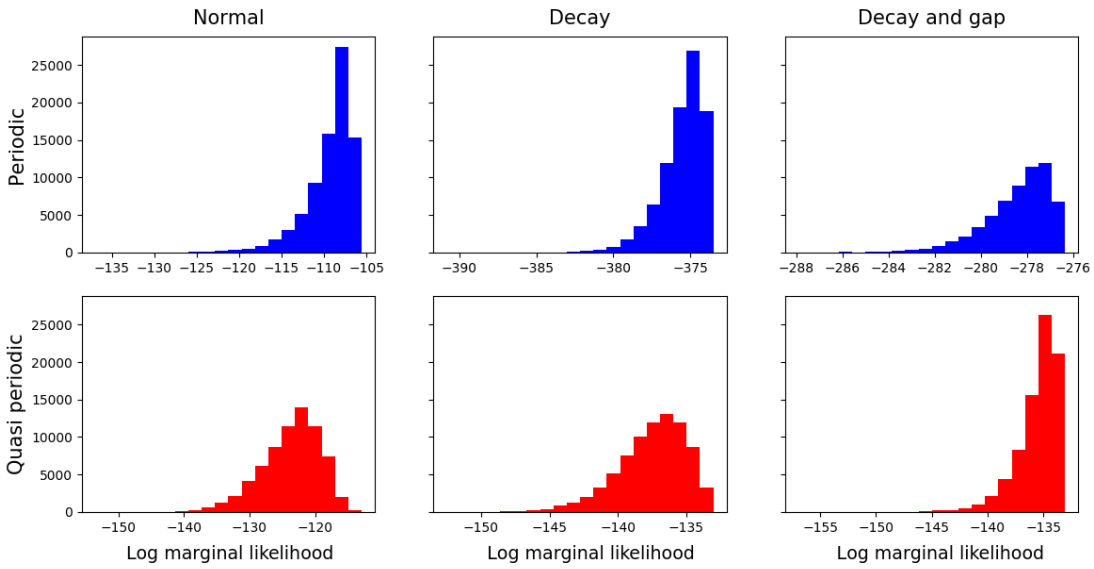


Figure 5.5: Log marginal likelihood obtained for each kernel while analyzing the RV signal generated by ten spots having a measurement per day.

Changing to the datasets with one measurement every four days, we obtain once again a very low estimation for the length-scale of the periodic kernel with the dataset with a linear decay. In table 5.11 we see that besides this low length-scale we also obtain a estimation of the period whose relative error is around 4.2%, far higher than the two other estimations of 0.2% and 0.3%, for the data without decay and the data with linear decay and a gap, respectively. This higher relative error was also observed in the analysis of one and five spots, which made us believe that when the estimation of a smaller length-scale occurs, it influences the estimation of the period increasing its relative error as a consequence.

On the other hand, the estimation of the period with the quasi-periodic kernel had relative errors around the 0.01% and 0.3% in the three cases, and thus a better estimation of the period when using the dataset containing a linear decay (table 5.12). Once again the white noise was far underestimated, most likely having been absorbed either by the estimation of the amplitude or the length-scale 2.

Table 5.11: MCMC results for the analysis of ten spots with the periodic kernel while having a measurement every 4 days, and having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$64.705^{+27.331}_{-16.898}$	$47.632^{+11.275}_{-8.328}$	$42.255^{+23.895}_{-10.467}$
Length-scale	$0.934^{+0.110}_{-0.093}$	$0.123^{+0.100}_{-0.081}$	$0.632^{+0.520}_{-0.289}$
Period (days)	$25.001^{+0.005}_{-0.005}$	$24.009^{+0.196}_{-0.208}$	$25.139^{+0.175}_{-0.224}$
White noise (m/s)	$0.04^{+0.078}_{-0.004}$	$0.005^{+0.110}_{-0.005}$	$10.369^{+2.941}_{-9.753}$

Table 5.12: MCMC results for the analysis of ten spots with the quasi-periodic kernel while having a measurement every 4 days, and having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$56.774^{+23.210}_{-13.993}$	$71.879^{+38.271}_{-21.233}$	$63.590^{+39.057}_{-19.720}$
Length-scale 1	$0.841^{+0.123}_{-0.082}$	$1.136^{+0.223}_{-0.166}$	$1.060^{+0.252}_{-0.198}$
Length-scale 2 (days)	$4966.772^{+5220.334}_{-2331.383}$	$216.522^{+69.994}_{-51.350}$	$208.456^{+92.763}_{-60.283}$
Period (days)	$25.004^{+0.008}_{-0.006}$	$25.121^{+0.088}_{-0.109}$	$25.054^{+0.099}_{-0.098}$
White noise (m/s)	$0.005^{+0.166}_{-0.004}$	$0.004^{+0.064}_{-0.004}$	$0.010^{+0.330}_{-0.009}$

Even so when the calculation of the log marginal likelihood was made to check which kernels would fit better, the quasi-periodic kernel had again a better performance of the two kernels, as seen in figure 5.6. This is important since, of the two kernels, the periodic kernel have obtained a worst estimation of the period, making it less reliable to determine the rotation period of the star for this cases.

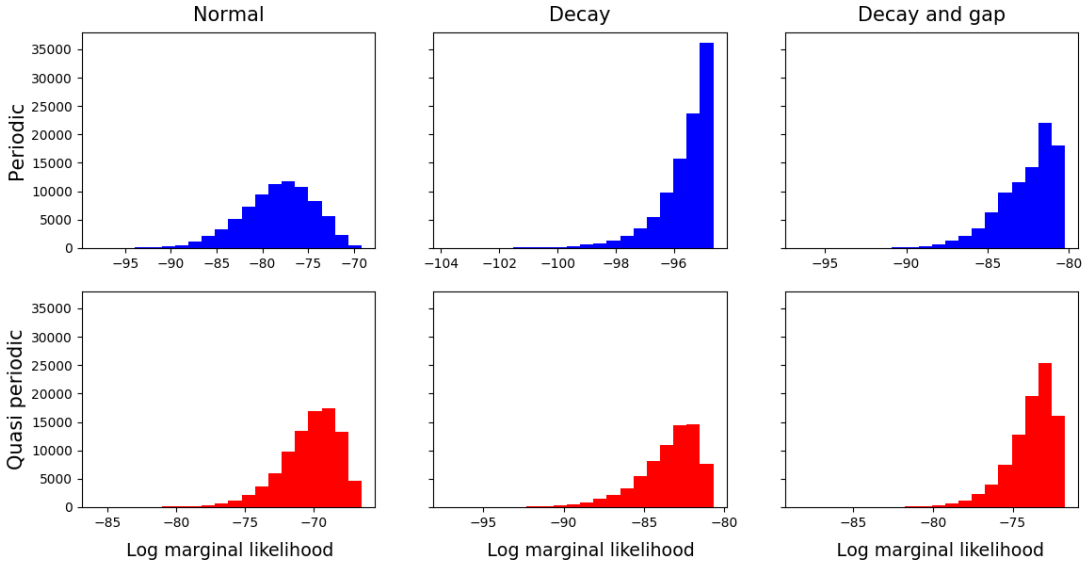


Figure 5.6: Log marginal likelihood obtained for each kernel while analyzing the RV signal of ten spots with a measurement every 4 days.

5.1.4 Twenty spots analysis

The last round of analysis performed was made in a SOAP dataset containing 20 randomly distributed spots. As in the previous analyses we added white noise, then a linear decay, and lastly a gap of thirty days, to allow three different analysis with each kernel.

When using the periodic kernel (table 5.13) and the quasi-periodic kernel (table 5.14), with the datasets with a measurement per day, the estimated period once again converged to a value very close to the real value of 25.05 days, as in all previous tests. In all cases the relative error of the estimated period was around 0.2%. Another estimation very similar on both kernels was seen in the amplitude, were for each the three datasets, both seemed to converge to similar values.

Although the white noise estimations in the dataset with no decay was similar in the two kernels, when the decay and the decay and a gap were added, the quasi-periodic kernel continued

Table 5.13: MCMC results for the analysis of twenty spots with the periodic kernel while having a measurement per day, and having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$48.336^{+12.936}_{-9.395}$	$68.100^{+48.556}_{-23.370}$	$64.795^{+46.778}_{-22.362}$
Length-scale	$0.579^{+0.032}_{-0.030}$	$0.960^{+0.274}_{-0.192}$	$0.956^{+0.280}_{-0.198}$
Period (days)	$25.000^{+0.001}_{-0.001}$	$24.998^{+0.038}_{-0.036}$	$24.995^{+0.048}_{-0.048}$
White noise (m/s)	$0.001^{+0.011}_{-0.001}$	$7.897^{+0.600}_{-0.574}$	$9.133^{+0.918}_{-0.786}$

Table 5.14: MCMC results for the analysis of twenty spots with the quasi-periodic kernel while having a measurement per day, and having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$43.079^{+8.137}_{-7.201}$	$69.888^{+23.713}_{-15.931}$	$65.932^{+27.907}_{-15.525}$
Length-scale 1	$0.577^{+0.033}_{-0.032}$	$0.678^{+0.045}_{-0.039}$	$0.674^{+0.050}_{-0.042}$
Length-scale 2 (days)	$17800.378^{+2978.164}_{-4938.85}$	$363.203^{+102.248}_{-76.263}$	$337.016^{+115.773}_{-76.523}$
Period (days)	$25.000^{+0.001}_{-0.001}$	$25.004^{+0.017}_{-0.017}$	$25.009^{+0.026}_{-0.025}$
White noise (m/s)	$0.001^{+0.012}_{-0.001}$	$0.002^{+0.026}_{-0.002}$	$0.005^{+0.063}_{-0.004}$

to underestimate the white noise, while the periodic kernel estimations increased to values an order of magnitude higher, a situation also observed in previous tests. When estimating the aperiodic length-scale in table 5.14, the estimation when using the SOAP dataset with noise added converge to a very high value, as expected, decreasing the impact of the aperiodic term in the kernel. In the other two datasets such behavior, again as expected, was not observed.

As done in all the previous analysis, the log marginal likelihood had to be used to determine what kernel was fitting better to the analyzed data. In figure 5.7, we can see that the histograms obtained once again showed that in the datasets with a linear decay, and with a linear decay and a gap, we were not working with strictly periodic data, thus making the periodic kernel to have a worse fit, while in the dataset we had no decay and gaps the periodic kernel performed slightly better.

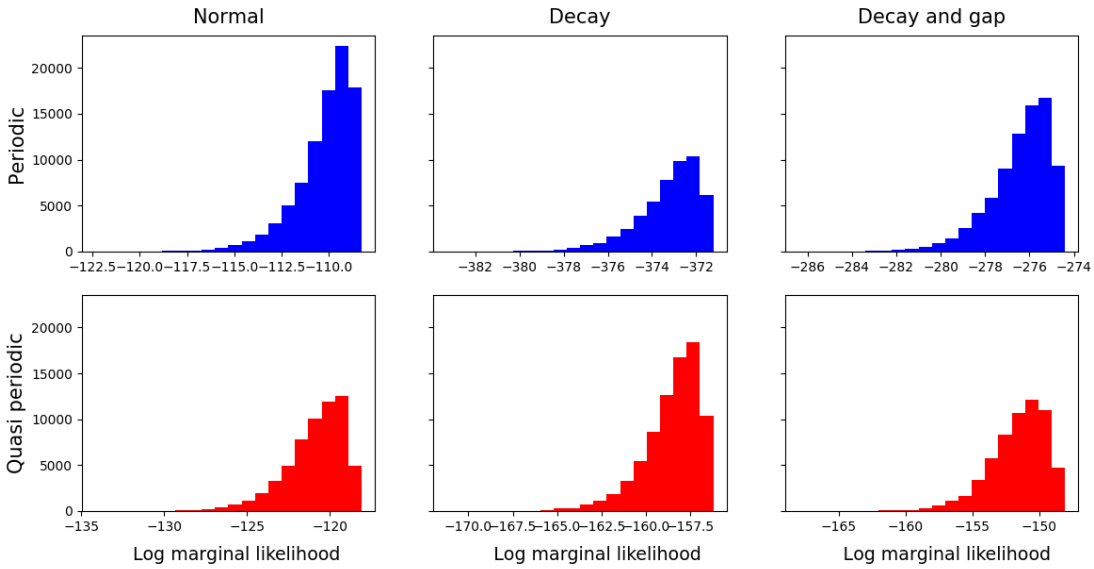


Figure 5.7: Log marginal likelihood obtained for each kernel while analyzing the RV signal generated by twenty spots having a measurement per day.

When the analysis with the datasets with a measurement every four days was performed, the estimated results had a similar behavior as earlier analysis. In table 5.15 we can see the results for the periodic kernel where in can be seen that once again the length scale for the dataset with a linear decay decreased in comparison to the other two datasets' estimations and the relative error of the period was around 4.1%, instead of being between 0.2% and 0.5% as it happened to the datasets with no decay and a linear decay with a gap.

For the quasi-periodic kernel (table 5.8) the estimated parameters behaved similarly as previous analysis. The period again reach a value whose relative error could be calculated as being between 0.1% and 0.5%, while the length-scale 2 in the analysis of the dataset without linear decay was a value that seemed to tend very high values, the following two analysis saw it decrease considerably.

Table 5.15: MCMC results for the analysis of twenty spots with the periodic kernel while having a measurement every 4 days, and having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$59.165^{+21.509}_{-14.061}$	$49.490^{+10.004}_{-8.207}$	$43.515^{+19.307}_{-9.929}$
Length-scale	$0.692^{+0.132}_{-0.081}$	$0.118^{+0.138}_{-0.103}$	$0.526^{+0.195}_{-0.171}$
Period (days)	$25.010^{+0.009}_{-0.008}$	$24.002^{+0.297}_{-0.289}$	$25.185^{+0.179}_{-0.175}$
White noise (m/s)	$0.257^{+0.542}_{-0.256}$	$0.004^{+0.170}_{-0.003}$	$7.548^{+3.225}_{-7.057}$

Table 5.16: MCMC results for the analysis of twenty spots with the quasi-periodic kernel while having a measurement every 4 days, and having the same conditions as of table 5.1.

	Normal	Decay	Decay and gap
Amplitude (m/s)	$58.452^{+20.129}_{-12.583}$	$60.574^{+26.432}_{-15.939}$	$47.295^{+20.247}_{-11.399}$
Length-scale 1	$0.721^{+0.064}_{-0.060}$	$0.824^{+0.167}_{-0.138}$	$0.622^{+0.157}_{-0.154}$
Length-scale 2 (days)	$10751.618^{+6939.236}_{-5702.923}$	$145.771^{+51.666}_{-34.269}$	$158.927^{+95.604}_{-51.326}$
Period (days)	$25.004^{+0.007}_{-0.007}$	$24.926^{+0.106}_{-0.133}$	$25.026^{+0.149}_{-0.131}$
White noise (m/s)	$0.004^{+0.078}_{-0.004}$	$0.004^{+0.124}_{-0.004}$	$0.026^{+2.470}_{-0.026}$

Since as in the previous tests the estimated values did not show any unexpected value in the hyperparameters determined, it was calculated the log marginal likelihood with the data from the MCMC to observe what kernel would fit the data better. Here again the quasi-periodic kernel fits the data better in the three cases, including the one with strictly periodic behavior, allowing us to conclude that this kernel, as it was expected, is a better choice when analysis this type of data.

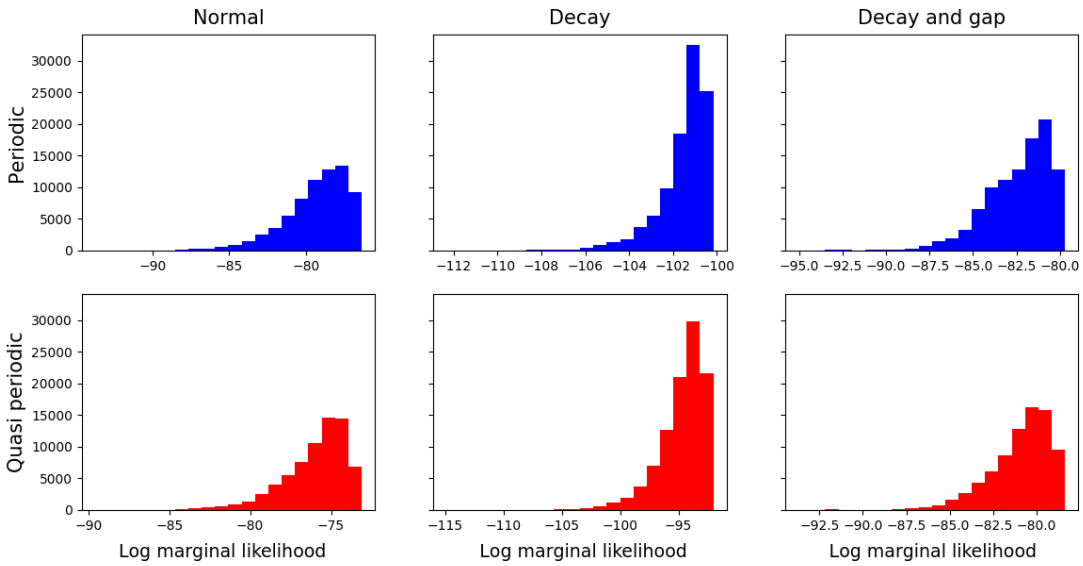


Figure 5.8: Log marginal likelihood obtained for each kernel while analyzing the RV signal of twenty spots with a measurement every 4 days.

5.1.5 Radial velocity measurements of spots and a planet

The last analysis done in this chapter consisted in trying to detect the radial velocity signal of a planet in a dataset containing both planets and spots radial velocity measurements. With that in mind, to the dataset generated by SOAP for five spots with a linear decay to simulate a simplistic form of spot evolution, we added the signal of a planet with a mass equal to 0.25 $M_{Jupiter}$, generating a radial velocity semi-amplitude (K) of 16.343 m/s, an orbital period (P_{planet}) of 30 days and an orbital eccentricity (e) of 0.6.

The radial velocity measurements of this new dataset, that can be seen in figure 5.9, were then analyzed using *gedi* and *emcee*. This time, instead of considering a zero mean function in our Gaussian process, was used a Keplerian function (equation 3.6) capable of interpreting the radial-velocity measurements caused by a planet in a Keplerian orbit around our star, and thus, capable of estimating the radial velocity semi-amplitude, period, and eccentricity of our planet.

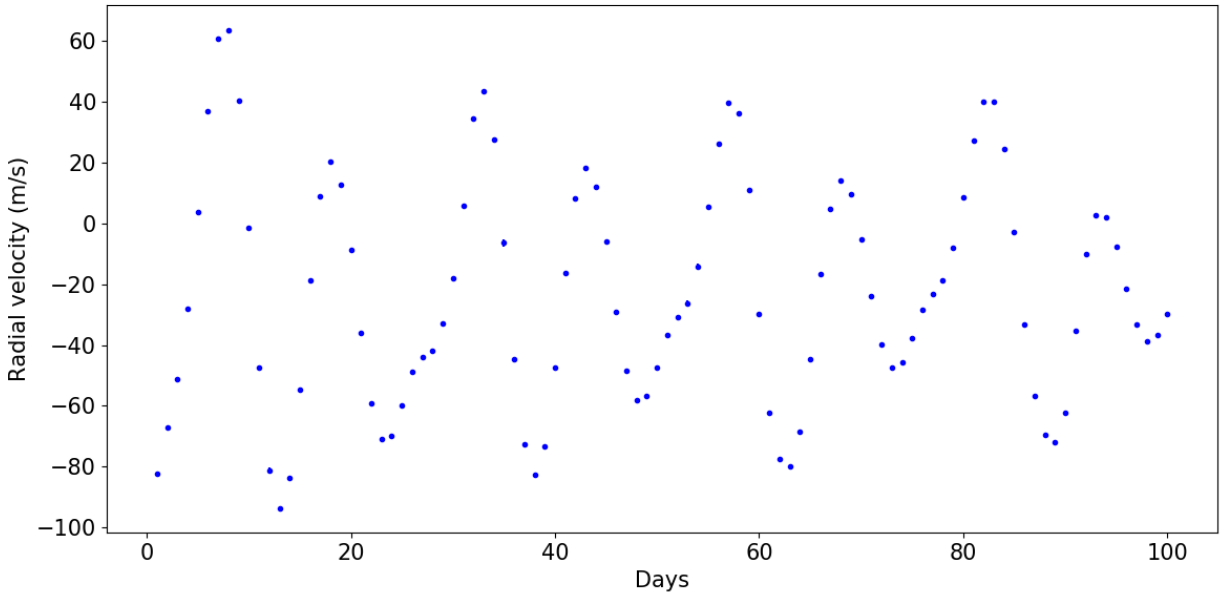


Figure 5.9: Radial velocity measurements of the dataset created using the SOAP dataset of five spots with a linear decay, and the signal of a planet orbiting the star. Although not noticeable, the data also includes the error bars in our measurements.

Using an MCMC with the periodic kernel first we obtained the following results. As seen in table 5.17, the periodic kernel was capable of estimating the star's rotation period with a relative error around 0.2%, showing a behavior similar to the one obtained in the previous analysis of the datasets with just starspots radial velocity measurements.

Table 5.17: MCMC results for the analysis of the dataset containing the radial velocity measurements of five spot with a linear decay, and a planet with $0.25 M_{Jupiter}$, in a 30 days orbit with, and eccentricity of 0.6. Using a periodic kernel plus white noise and a Keplerian function as mean function.

	Estimation
Kernel amplitude (m/s)	$75.236^{+53.547}_{-27.993}$
Length-scale	$1.021^{+0.226}_{-0.217}$
Kernel period (days)	$24.999^{+0.044}_{-0.046}$
White noise (m/s)	$8.579^{+0.669}_{-0.569}$
P_{planet} (days)	$30.089^{+0.158}_{-0.247}$
e	$0.604^{+0.118}_{-0.135}$
K (m/s)	$13.235^{+2.960}_{-2.393}$

With the Keplerian function it was also possible to obtain a fairly good estimation of the planet's parameters. The period of he planet, radial velocity semi-amplitude, and eccentricity were estimated with relative errors around 0.2%, 17.3%, and 0.7%, when compared with the real values. Although the value of K was the more distant for its real counterpart, when considering the error interval of this estimation the real value falls into this range.

With the quasi-periodic kernel it was not possible to obtain convergence in the most important parameters that were being estimated, and thus the results are not presented here. This might have occured due to hardware limitations and a solution would be to set a larger number of iterations in the MCMC but that was not possible.

Chapter 6.
Conclusions and future work

"See you in about 30 years."

Emmett Brown (1920-????)

From the analysis performed in this thesis, it can be seen that Gaussian processes can be used with success in the analysis of radial velocity measurements. With all the available data from HARPS and the future data that will be collected by ESPRESSO, Gaussian processes are seen as a powerful tool to use, and the package developed in this thesis can help in the analysis of all future data.

When compared with other existing packages it is seen that *gedi*, the python package developed in this thesis, showed to have a good performance. Although when *gedi* was used with *scipy* it showed that it had a worse performance in comparison to *george*, the tests using *emcee* made us conclude that our package is a good substitute when it becomes necessary to use Gaussian processes with an MCMC.

Our analysis with just randomly distributed spots showed that, while both periodic and quasi-periodic kernels were capable of obtaining a good estimation of the kernel's hyperparameters, the quasi-periodic kernel log marginal likelihood has shown that this kernel fits better in all observed cases where we have simulated spot evolution and gaps in the data. This seems to confirm that the quasi-periodic kernel is, of the two in analysis in this thesis, the best to use in radial velocity measurements when we want to detect or confirm the rotational period of the star.

When the objective was to detect the radial velocity signal of a planet in a dataset containing both planets and the activity signal due to spots, unlike the previous analysis, the periodic kernel behaved better, as it was capable of estimating the planets parameters while the quasi-

periodic kernel was not capable of obtaining a satisfactory result in its hyperparameters and futher tests to this kernel will be needed in a future work.

6.1 Future work

Not done in this thesis, the next step would be to use *gedi* and the properties that it has implemented is of course in real available data, for example, by HARPS. While this thesis shows that we are capable of estimating the period of rotation of a star, and the planetary properties when a planet is included in the dataset, the data was generated having into consideration only the noise caused by starspots, and its evolution treated in a simplistic way. Thus the analysis of real data will contain a more complex stellar noise on it that will require a careful set of tests to determine if *gedi* is capable of dealing with it.

If *gedi* proves to be a good package to work with Gaussian processes and more importantly, proves to be successful in interpreting real planetary signal and successful estimate its most important parameters, it can be a tool to have into consideration when ESPRESSO data becomes available in the next couple of years.

Bibliography

Abramowitz, M. and I. Stegun (1972). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. 10th. National Bureau of Standards. ISBN: 0-486-61272-4.

Adibekyan, V., P. Figueira, and N. C. Santos (2016). *Which Type of Planets do We Expect to Observe in the Habitable Zone?* Origins of Life and Evolution of the Biosphere 46, 351-359. DOI: 10.1007/s11084-016-9486-1.

Ambikasaran, S. et al. (2014). "Fast Direct Methods for Gaussian Processes and the Analysis of NASA Kepler Mission Data". In: *ArXiv*. eprint: 1403.6015.

Barros, S. C. C. et al. (2013). *Transit timing variations in WASP-10b induced by stellar activity*. MNRAS 430, 3032-3047. DOI: 10.1093/mnras/stt111.

Blum, M. and M. Riedmiller (2013). *Optimization of Gaussian Process Hyperparameters using RPROP*. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.

Brewer, B. J. and D. Stello (2009). *Gaussian Process Modelling of Asteroseismic Data*. MNRAS 395, 2226-2233. DOI: 10.1111/j.1365-2966.2009.14679.x.

Butler, R. P. and G. W. Marcy (1996). *A Planet Orbiting 47 Ursae Majoris*. Astrophysical Journal Letters 464, L153. DOI: 10.1086/310102.

Charbonneau, D. et al. (1999). *Detection of Planetary Transits Across a Sun-Like Star*. Astrophys. J. 529, L45-L48. DOI: 10.1086/312457.

Csató, L. and M. Opper (2001). *Sparce Online Gaussian Processes*. Neural Computation 14 (3), 641-668. Aston University. DOI: 10.1162/089976602317250933.

Do, C. B. (2008). *CS229 Machine Learning Lecture Notes*. Stanford University.

Dumusque, X., I. Boisse, and N. C. Santos (2014). *SOAP 2.0: A Tool to Estimate the Photometric and Radial Velocity Variations Induced by Stellar Spots and Plages*. ApJ 796, 132. DOI: 10.1088/0004-637X/796/2/132.

Dumusque, X. et al. (2011a). *Planetary detection limits taking into account stellar noise. I. Observational strategies to reduce stellar oscillation and granulation effects.* A&A 525, A140. DOI: 10.1051/0004-6361/201014097.

Dumusque, X. et al. (2011b). *Planetary detection limits taking into account stellar noise. II. Effect of stellar spot groups on radial-velocities.* A&A 527, A82. DOI: 10.1051/0004-6361/201015877.

Dumusque, X. et al. (2012). *An Earth-mass planet orbiting Alpha Centauri B.* Nature 491, 207-211. DOI: 10.1038/nature11572.

Duvenaud, D. K. (2014). *Automatic Model Construction with Gaussian Processes.* PhD Thesis. University of Cambridge.

Faria, J. P. et al. (2016). *Uncovering the Planets and stellar activity of CoRoT-7 using only radial velocities.* A&A 588, A31. DOI: 10.1051/0004-6361/201527899.

Fischer, D. A. et al. (2016). *State of the Field: Extreme Precision Radial Velocities.* Publications of the Astronomical Society of the Pacific 128(964). DOI: 10.1088/1538-3873/128/964/066001.

Foreman-Mackey, D. et al. (2013). *emcee: The MCMC Hammer.* PASP 125, 306. DOI: 10.1086/670067.

Goodman, J. and J. Weare (2010). *Ensemble samplers with affine invariance.* Applied Mathematics and Computational Science 5, 65-80. DOI: 10.2140/camcos.2010.5.65.

Grunblatt, S. K., A. W. Howard, and R. D. Haywood (2015). *Determining the Mass of Kepler-78b with Nonparametric Gaussian Process Estimation.* ApJ 808, 127. DOI: 10.1088/0004-637X/808/2/127.

Hamra, G., R. MacLehose, and D. Richardson (2013). *Markov Chain Monte Carlo an introduction for epidemiologists.* International Journal of Epidemiology 42, 627-634. DOI: 10.1093/ije/dyt043.

Hatzes, A. P. et al. (2010). *An Investigation into the Radial Velocity Variations of CoRoT-7.* A&A 520, A93. DOI: 10.1051/0004-6361/201014795.

Haywood, R. D. et al. (2014). *Planets and stellar activity: hide and seek in CoRoT-7 system.* MNRAS 443, 2517-2531. DOI: 10.1093/mnras/stu1320.

Henry, G. W. et al. (2000). *A Transiting "51 Peg-like" Planet*. ApJ 529, 1. DOI: 10.1086/312458.

Horn, R. A. and C. R. Johnson (2013). *Matrix Analysis*. 2nd. Cambridge University Press. ISBN: 978-0-521-83940-2.

Jenkins, J. S. et al. (2013). *Two Super-Earths Orbiting the Solar Analog HD 41248 on the Edge of a 7:5 Mean Motion Resonance*. ApJ 771, 41. DOI: 10.1088/0004-637X/771/1/41.

Kriege, D. G. (1951). *A Statistical Approach to Some Mine Valuation and Allied Problems on the Witwatersrand*. MSc Thesis. University of the Witwatersrand.

Lagrange, A.-M., M. Desort, and N. Meunier (2010). *Using the Sun to estimate Earth-like planets detection capabilities. I. Impact of cold spots*. A&A 512, A38. DOI: 10.1051/0004-6361/200913071.

Léger, A. et al. (2009). *Transiting Exoplanets from the CoRoT Space Mission - VIII. CoRoT-7b: The First Super-Earth with Measured Radius*. A&A 506, 287. DOI: 10.1051/0004-6361/20091193.

Lifshits, M. (2012). *Lectures on Gaussian Processes*. 2nd. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-642-24938-9. DOI: 10.1007/978-3-642-24939-6.

Lin, D. N. C., P. Bodenheimer, and D. C. Richardson (1996). *Orbital migration of the planetary companion of 51 Pegasi to its present location*. Nature 380, 606-607. DOI: 10.1038/380606a0.

Marcy, G. W. and R. P. Butler (1996). *A Planetary Companion to 70 Virginis*. Astrophysical Journal Letters 464, L147. DOI: 10.1086/310096.

Matheron, G. (1962). *Traité de géostatistique appliquée*. Éditions Technip: Paris, NA.

Mayor, M., C. Lovis, and N. C. Santos (2014). *Dopler Spectroscopy as a Path to the Detection of Earth-like Planets*. Nature 513, 328-335. DOI: 10.1038/nature13780.

Mayor, M. and D. Queloz (1995). *A Jupiter-mass companion to a solar-type star*. Nature 378, 355-359. DOI: 10.1038/378355a0.

Mcleod, A. I. and B. Quenneville (2001). *Mean likelihood estimators*. Statistics and Computing 11, 57-65. DOI: 10.1023/A:1026509916251.

Meunier, N., M. Desort, and A.-M. Lagrange (2010). *Using the Sun to estimate Earth-like planets detection capabilities. II. Impact of plages*. A&A 512, A39. DOI: 10.1051/0004-6361/200913551.

Neal, R. M. (1998). *Regression and Classification Using Gaussian Process Priors*. Bayesian Statistics 6 475-501. Oxford University Press.

Neumann, M. et al. (2015). *pyGPs - A Python Library for Gaussian Process Regression and Classification*. Journal of Machine Learning Research 16, 2611-2616.

Nocedal, J. and S. J. Wright (2006). *Numerical Optimization*. 2nd. Springer-Verlag New York. ISBN: 978-0-387-40065-5.

Orloff, J. and J. Bloom (2014). *18.05 Introduction to Probability and Statistics*. Massachusetts Institute of Technology: MIT OpenCourseWare.

Pepe, F., M. Mayor, and G. Rupprecht (2002). *HARPS: ESO's Coming Planet Searcher*. The Messenger 110, 9-14.

Pepe, F. et al. (2013). *ESPRESSO - An Echelle SPectrograph for Rocky Exoplanets Search and Stable Spectroscopic Observations*. The Messenger 153, 6-16.

Perryman, M. (2011). *The Exoplanet Handbook*. 1st. Cambridge University Press. ISBN: 978-0-521-76559-1.

Queloz, D. et al. (2009). *The CoRoT-7 Planetary System: Two Orbiting Super-Earths*. A&A 506, 303. DOI: 10.1051/0004-6361/200913096.

Rajpaul, V. et al. (2015). *A Gaussian Process framework for modelling stellar actitivty signals in radial velocity data*. MNRAS 452, 2269-2291. DOI: 10.1093/mnras/stv1428.

Rao, S. S. (2009). *Engineering Optimization: Theory and Practiceg*. 4th. John Wiley and Sons Inc. ISBN: 978-0-470-18352-6.

Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian Processes for Machine Learning*. 1st. MIT Press. ISBN: 0-262-18253-X.

Rice, K. (2014). *The Detection and Characterization of Extrasolar Planets*. Challenges 5, 296-323. DOI: 10.3390/challe5020296.

Riedmiller, M. and H. Braun (1993). *A direct adaptive method for faster back-propagation learning: the RPROP algorithm*. IEEE International Conference on Neural Network. DOI: 10.1109/ICNN.1993.298623.

Robert, C. and G. Casella (2004). *Monte Carlo Statistical Methods*. 2nd. Springer-Verlag New York. ISBN: 978-0-387-21239-5.

Saar, S. H. and R. A. Donahue (1997). *Activity-Related Radial Velocity Variation in Cool Stars*. ApJ 487, 319-327. DOI: 10.1086/304392.

Santos, N. et al. (2000). *The CORALIE survey for Southern extra-solar planets IV. Intrinsic stellar limitations to planet searches with radial-velocity techniques*. A&A 361, 265-272. DOI: 10.1051/0004-6361:20020876.

Santos, N. C. et al. (2014). *The HARPS search for southern extra-solar planets XXXV. The interesting case of HD 41248: stellar activity, no planets?* A&A 566, A35. DOI: 10.1051/0004-6361/201423808.

Sivia, D. and J. Skilling (2006). *Data analysis: a Bayesian tutorial*. Oxford science publications. Oxford University Press. ISBN: 9780198568315.

Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. 1st. Springer-Verlag New York. ISBN: 978-0-387-98629-6.

Struve, O. (1952). *Proposal for a project of high-precision stellar radial velocity work*. The Observatory 72, 199-200.

Titsias, M. K., M. Rattray, and N. D. Lawrence (2011). *Markov chain Monte Carlo algorithms for Gaussian processes*. 1st. Bayesian Time Series Models (pp. 295-316). Cambridge University Press. DOI: 10.1017/CBO9780511984679.015.

Udry, S. and N. C. Santos (2007). *Statistical Properties of Exoplanets*. Annu. Rev. Astron. Astrophys. 45, 397-439. DOI: 10.1146/annurev.astro.45.051806.110529.

Williams, C. K. (1998). *Prediction with Gaussian processes: From linear regression to linear prediction and beyond*. Learning in graphical models, pp 599-621. Springer Netherlands. DOI: 10.1007/978-94-011-5014-9_23.

Williams, C. K. and D. Barber (1998). *Bayesian Classification with Gaussian Processes*. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(12) 1342-1351. DOI: 10.1109/34.735807.

Wilson, A. G. (2014). *Covariance Kernels for Fast Automatic Pattern Discovery and Extrapolation with Gaussian Processes*. PhD Thesis. University of Cambridge.

Wright, J. T. and B. S. Gaudi (2013). *Exoplanet Detection Method*. 1st. Planets, Stars and Stellar Systems. Springer Netherlands. DOI: 10.1007/978-94-007-5606-9_10.

Appendices

Chapter A.
Algorithms for optimization

In this section we present a simple pseudo-code of the algorithms discussed in chapter 2 that were implemented in the *gedi* package. A more detailed analysis of algorithms 1 and 2, including their deduction, can be seen in Nocedal and Wright (2006) and in Rao (2009).

Algorithm 1 Steepest descent algorithm

1: Given a initial point X_1 and stopping criteria to determine when the algorithm stops
2: Set $i = 1$
3: Find a search direction $S_i = -\nabla f(X_i)$
4: Determine optimal step length λ_i
5: Set $X_{i+1} = X_i + \lambda_i S_i$
6: Evaluate stopping criteria
7: **if** stopping criteria are not met **then**
8: i=i+1
9: Back to line 3
10: **if** stopping criteria are met **then**
11: Return X_{i+1}
12: **End algorithm**;

Algorithm 2 BFGS algorithm

1: Given a initial point X_1, a $n \times n$ positive defined matrix, B_1, a initial estimate of the inverse Hessian matrix, and stopping criteria to determine when the algorithm stops.
2: Set $i = 1$
3: Find search direction $S_i = -B_1 \nabla f(X_i)$
4: Determine optimal step length λ_i
5: Set $X_{i+1} = X_i + \lambda_i S_i$
6: Evaluate stopping criteria
7: **if** stopping criteria are not met **then**
8: Calculate $d_i = \lambda_i S_i$
9: Calculate $g_i = \nabla f(X_{i+1}) - \nabla f(X_i)$
10: Update the Hessian matrix $B_{i+1} = B_i + \left(1 + \frac{g_i^\intercal B_i g_i}{d_i^\intercal g_i}\right) \frac{d_i d_i^\intercal}{d_i^\intercal g_i} - \frac{d_i g_i^\intercal B_i}{d_i^\intercal g_i} - \frac{B_i g_i d_i^\intercal}{d_i^\intercal g_i}$
11: i=i+1
12: Back to line 3
13: **if** stopping criteria are met **then**
14: Return X_{i+1}
15: **End algorithm**;

Algorithm 3 Alternative steepest descent algorithm

1: Given a initial point X_1, a step length λ_1, and stopping criteria to determine when the algorithm stops
2: Set $i = 1$
3: Find a search direction $S_i = -\nabla f(X_i)$
4: Set $X_{i+1} = X_i + \lambda_i S_i$
5: **if** $\nabla f(X_i)$ and $\nabla f(X_{i+1})$ have the same sign **then**
6: $\quad \lambda_{i+1} = 1.2\lambda_i$
7: $\quad X_{i+1} = X_{i+1}$
8: **if** $\nabla f(X_i)$ and $\nabla f(X_{i+1})$ do not have the same sign **then**
9: $\quad \lambda_{i+1} = 0.5\lambda_i$
10: $\quad X_{i+1} = X_i$
11: Evaluate stopping criteria
12: **if** stopping criteria are not met **then**
13: $\quad$ i=i+1
14: $\quad$ Back to line 3
15: **if** stopping criteria are met **then**
16: $\quad$ Return X_{i+1}
17: **End algorithm**;

Chapter B.
Gedi examples

In this section we present two simple examples of how to work with *gedi*. In the first it will be presented a simple example on how to perform the optimization of the kernel with *scipy.optimize*. The second one will be another simple example but this time using the package *emcee*. With this two example we believe we are able to show the full potential of *gedi* and of the functions it has implemented.

B.1 scipy.optimize

To use *gedi* together with *scipy.optimize* we first we of course to import all the necessary python packages. Besides the two mentioned packages we will also require *numpy*.

```
import numpy as np
import Gedi as gedi
import scipy.optimize as op
```

Having imported all the necessary packages we can now simulate some some sinusoidal data.

```
np.random.seed(1001)
x= 10 * np.sort(np.random.rand(30))
yerr= 0.5 * np.ones_like(x)
y= np.sin(x) + yerr * np.random.randn(len(x))
pl.plot(x,y,'*')
```

To have some consistency in the examples showed in this appendix we also define a seed to allow the reproduction of the results in the future.

With the generated data we can now choose a kernel to use with it and calculate the respective log marginal likelihood

```
#first kernel
kernel0= gedi.kernel.ExpSineSquared(2,2.5,5)
kernel_lk0= gedi.kernel_likelihood.likelihood(kernel0,x,y,yerr)
print('initial likelihood',kernel_lk0)

#second kernel
kernel= gedi.kernel.ExpSineSquared(2,2.5,5) + gedi.kernel.WhiteNoise(0.2)
kernel_lk= gedi.kernel_likelihood.likelihood(kernel,x,y,yerr)
print('initial likelihood',kernel_lk)
```

Since we are working with a sinusoid it is logical to use the *ES kernel* as it is a periodic kernel. From the first kernel we were able to obtain a log marginal likelihood of around -51.06, while with the second we obtained around -47.97. This clearly tell us that the second kernel is a better choice to work with, what comes with not much surprise, as the generated data contained noise.

As such we will continue our analysis using the second kernel and now define the log marginal likelihood and the gradients that *scipy.optimize* will use

```
#Log marginal likelihood
def likelihood_gedi(p):
    global kernel
    # Update the kernel parameters and compute the likelihood.
    kernel= gedi.kernel_optimization.new_kernel(kernel,np.exp(p))
    ll = gedi.kernel_likelihood.likelihood(kernel,x,y,yerr)
    return -ll if np.isfinite(ll) else 1e25

#Gradients
def gradients_gedi(p):
    global kernel
    # Update the kernel parameters and compute the likelihood.
    kernel= gedi.kernel_optimization.new_kernel(kernel,np.exp(p))
    return -np.array(gedi.kernel_likelihood.gradient_likelihood(kernel,x,y,yerr))
```

With this two simple functions we are now ready to use *scipy.optimize* and find our optimized kernel

```
#lets run the optimization
p0_gedi = np.log(kernel.pars)
results_gedi = op.minimize(likelihood_gedi, p0_gedi, jac=gradients_gedi)
```

```
kernel= gedi.kernel_optimization.new_kernel(kernel,np.exp(results_gedi.x))


print('Final kernel',kernel)
print('Final likelihood =',gedi.kernel_likelihood.likelihood(kernel,x,y,yerr))
```

This allow us to obtain as a final result

```
('Final kernel', ExpSineSquared(2.72896536025, 1.10953213369, 16.5826072023)
+ WhiteNoise(0.356778865898))
('Final likelihood', -34.288489695783142)
```

The final log marginal likelihood show us that the final kernel is indeed a better kernel than the one used in the beginning, and *scipy.optimize* was successful in finding a better solution to the one we had.

B.2 emcee

We are now going to use *gedi* in conjunction with *emcee* to find the best values of our kernel's hyperparameters. We obviously begin by importing all necessary packages

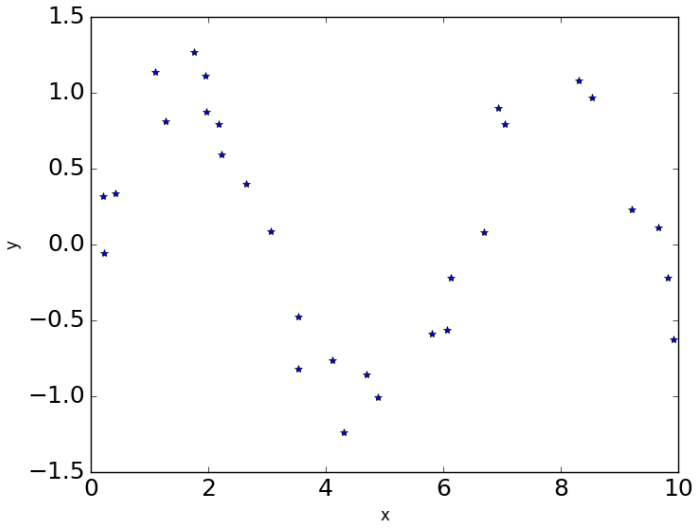
```
import Gedi as gedi
import emcee
import numpy as np
import matplotlib.pyplot as pl
from matplotlib.ticker import MaxNLocator
from scipy import stats
```

Besides *gedi* and *emcee* we will also need to use *numpy*, *scipy* and *matplotlib*, to generate our data and plot the necessary graphics. To start this second example we can generate same sinusoidal data with and respective error with

```
np.random.seed(1001)
x= 10 * np.sort(np.random.rand(30))
yerr= 0.2 * np.ones_like(x)
y= np.sin(x) + yerr * np.random.randn(len(x))
```

With the help of *matplotlib* we can take a look on our generated data.

```
pl.plot(x,y,'*')
pl.xlabel('x')
pl.ylabel('y')
```



This allow us to see that the amplitude of our data is around 1 unit and the period around 6 units, which will be useful to set our priors. Having prepared the data that we will be working with we can now prepare our MCMC and start by defining our priors and the log marginal likelihood we will use

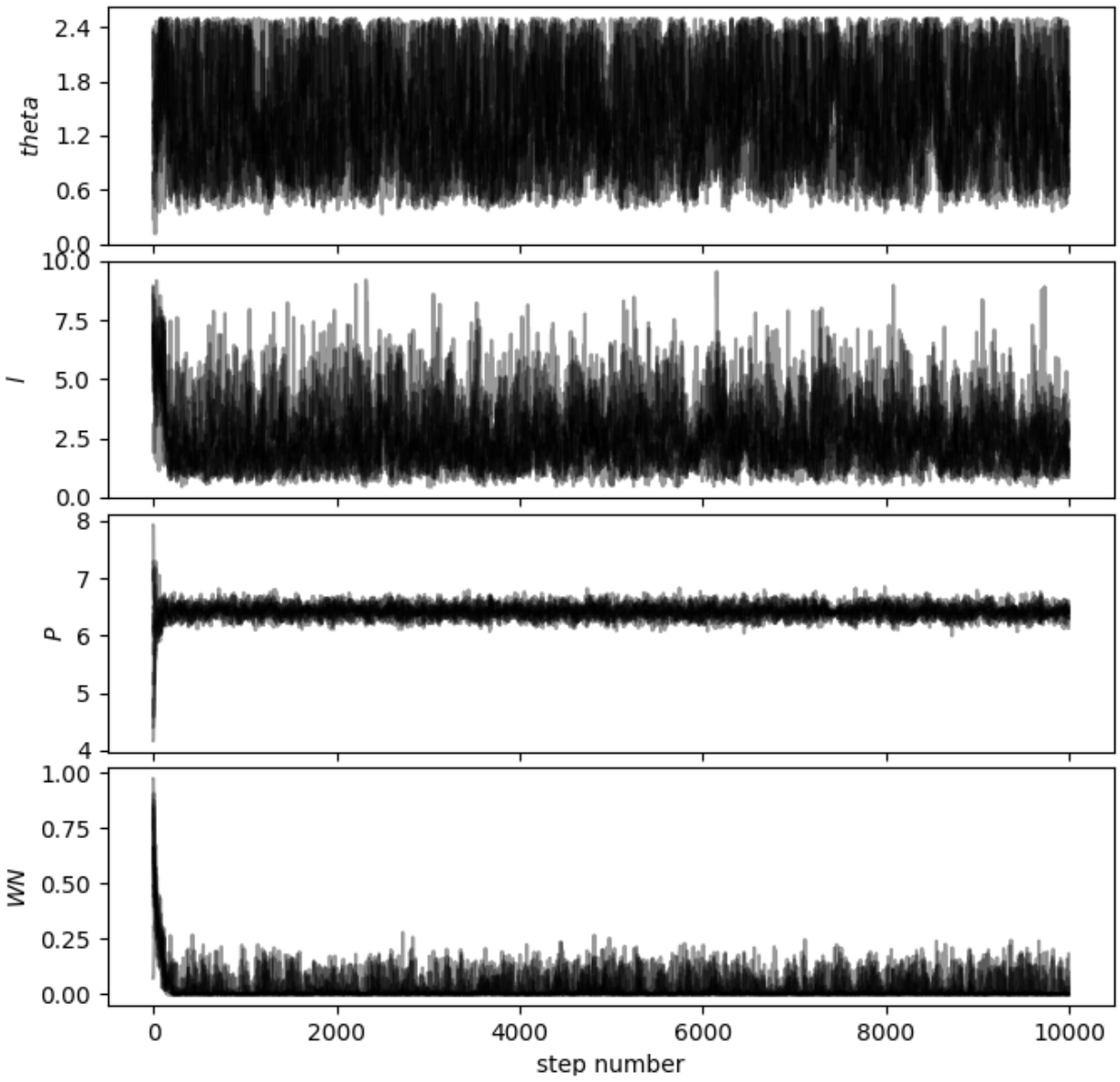
```
#defining our priors
def logprob(p):
    global kernel
    if any([p[0] < np.log(1), p[0] > np.log(2),
            p[1] < -10, p[1] > np.log(10),
            p[2] < np.log(4), p[2] > np.log(8),
            p[3] < -10, p[3] > np.log(0.5)]):
        return -np.inf
    logprior=0.0
    # Update the kernel and compute the log marginal likelihood.
    kernel=gedi.kernel_optimization.new_kernel(kernel,np.exp(p))
    new_likelihood=gedi.kernel_likelihood.likelihood(kernel,x,y,yerr)
    return logprior + new_likelihood

amplitude_prior=stats.uniform(1, 2-1)
lenghtscale_prior=stats.uniform(np.exp(-10), 10-np.exp(-10))
period_prior=stats.uniform(4, 8-4)
```

```
wn_prior=stats.uniform(np.exp(-10), 0.5-np.exp(-10))

def from_prior():
    return np.array([amplitude_prior.rvs(),lenghtscale_prior.rvs(),
                     period_prior.rvs(),wn_prior.rvs()])

#defining our kernel
kernel=gedi.kernel.ExpSineSquared(amplitude_prior.rvs(),
                lenghtscale_prior.rvs(),period_prior.rvs()) +\
                gedi.kernel.WhiteNoise(wn_prior.rvs())

#preparing our MCMC
burns, runs= 2500, 5000

#set up the sampler.
nwalkers, ndim = 10, len(kernel.pars)
sampler = emcee.EnsembleSampler(nwalkers, ndim, logprob)

p0=[np.log(from_prior()) for i in range(nwalkers)]
assert not np.isinf(map(lnprob, p0)).any()

p0, _, _ = sampler.run_mcmc(p0, burns)
sampler.run_mcmc(p0, runs)
```

Once again we use the sum of an *ES kernel* with a *WN kernel* to fit to our data, since our data comes from a sinusoidal model. With our MCMC complete we can now plot the results to check visually if we had convergence in our hyperparameters.

```
fig, axes = pl.subplots(4, 1, sharex=True, figsize=(8, 9))
axes[0].plot(sampler.chain[:, :, 0].T, color="k", alpha=0.4) #log
axes[0].yaxis.set_major_locator(MaxNLocator(5))
axes[0].set_ylabel("$theta$")
axes[1].plot(np.exp(sampler.chain[:, :, 1]).T, color="k", alpha=0.4)
axes[1].yaxis.set_major_locator(MaxNLocator(5))
axes[1].set_ylabel("$l$")
axes[2].plot(np.exp(sampler.chain[:, :, 2]).T, color="k", alpha=0.4)
axes[2].yaxis.set_major_locator(MaxNLocator(5))
axes[2].set_ylabel("$P$")
axes[3].plot(sampler.chain[:, :, 3].T, color="k", alpha=0.4) #log
```

```
axes[3].yaxis.set_major_locator(MaxNLocator(5))
axes[3].set_ylabel("$WN$")
axes[3].set_xlabel("step number")
fig.tight_layout(h_pad=0.0)
```



Using 5000 steps as our burn-in and 5000 step to use in our MCMC, we can see that there seems to be a convergence, for example, on the hyperparameter that corresponds to the period and the white noise of the kernel, although the same seemed to not be obtained to the amplitude and the length-scale, as this is just an example of how to use *gedi*, we will ignore it and continue on our analysis.

To obtain our final solution we can compute the quantiles and median that will be used

```
burnin = 50
samples = sampler.chain[:, burnin:, :].reshape((-1, ndim))


samples[:, 0] = np.exp(samples[:, 0])   #amplitude
samples[:, 1] = np.exp(samples[:, 1])   #lenght scale
samples[:, 2] = np.exp(samples[:, 2])   #period
samples[:, 3] = np.exp(samples[:, 3])   #white noise
theta_mcmc,l_mcmc,p_mcmc,wn_mcmc = map(lambda v: (v[1], v[2]-v[1], v[1]-v[0]),
                                zip(*np.percentile(samples, [16, 50, 84],
                                                   axis=0)))


print('theta = {0[0]} +{0[1]} -{0[2]}'.format(theta_mcmc))
print('l = {0[0]} +{0[1]} -{0[2]}'.format(l_mcmc))
print('period = {0[0]} +{0[1]} -{0[2]}'.format(p_mcmc))
print('white noise = {0[0]} +{0[1]} -{0[2]}'.format(wn_mcmc))
```

In the end of this we are able to obtain the values

```
theta = 1.41668244894 +0.373790095626 -0.288328791884
l = 2.46889155829 +0.952416345357 -0.735937463898
period = 6.42794862032 +0.096144793889 -0.0969172051794
white noise = 0.00225458063694 +0.0358745326978 -0.00209754174694
```

Which comparing with the value obtained in the optimization with scipy.optimize, using a MCMC has a greater advantage. Not only we were able to obtain a value for our hyperparameters, we were able to obtain an error interval for each one of it. Since the optimization of the hyperparameters with gradient based algorithms is not a convex problem, it is necessary to be careful about our initial values, in order to not reach a bad local minima.

The result obtained with *scipy.optimize* give us a period of around 16.58 units, while just by looking at the graph of the data analyzed, we should have a periodicity around 6 units. Unlike *scipy.optimize*, the MCMC clearly reach this value, showing us that the problem of bad local minima does not occur with the use of an MCMC.